AFRL-AFOSR-JP-TR-2016-0073

Large-scale Linear Optimization through Machine Learning: From Theory to Practical System Design and Implementation

Jinwoo Shin
Korea Advanced Institute of Science and Technology

08/10/2016
Final Report

Air Force Research Laboratory
AF Office Of Scientific Research (AFOSR)/ IOA
Arlington, Virginia 22203
Air Force Materiel Command

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
10-08-2016	Final	28 May 2014 to 27 May 2016

4. TITLE AND SUBTITLE
Large-scale Linear Optimization through Machine Learning: From Theory to Practical System Design and Implementation

5a. CONTRACT NUMBER

5b. GRANT NUMBER
FA2386-14-1-4058

5c. PROGRAM ELEMENT NUMBER
61102F

6. AUTHOR(S)
Jinwoo Shin

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Korea Advanced Institute of Science and Technology
291 Daehak-ro, Yuseong-gu
Taejon, 305701 KR

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AOARD
UNIT 45002
APO AP 96338-5002

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/AFOSR IOA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-AFOSR-JP-TR-2016-0073

12. DISTRIBUTION/AVAILABILITY STATEMENT
A DISTRIBUTION UNLIMITED: PB Public Release

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The linear programming (LP) is one of the most popular necessary optimization tool used for data analytics as well as in various scientific fields. However, the current state-of-art algorithms suffer from scalability issues when processing Big Data. For example, the commercial optimization software IBM CPLEX cannot handle an LP with more than hundreds of thousands variables or constraints. Existing algorithms are fundamentally hard to scale because they are inevitably too complex to parallelize. To address the issue, we study the possibility of using the Belief Propagation (BP) algorithm as an LP solver. BP has shown remarkable performances on various machine learning tasks and it naturally lends itself to fast parallel implementations. Despite this, very little work has been done in this area. In particular, while it is generally believed that BP implicitly solves an optimization problem, it is not well understood under what conditions the solution to a BP converges to that of a corresponding LP formulation.

Our efforts consist of two main parts. First, we perform a theoretic study and establish the conditions in which BP can solve LP [1,2]. Although there has been several works studying the relation between BP and LP for certain instances, our work provides a generic condition unifying all prior works for generic LP. Second, utilizing our theoretical results, we develop a practical BP-based parallel algorithms for solving generic LPs, and it shows 71x speed up while sacrificing only 0.1% accuracy compared to the state-of-art exact algorithm.

As a result of the study, the PIs have published two conference papers and two follow-up journal papers are under submission. We refer the readers to our published work for details.

15. SUBJECT TERMS
Cloud, AOARD

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			LUTZ, BRIAN
Unclassified	Unclassified	Unclassified	SAR	13	19b. TELEPHONE NUMBER *(Include area code)* 315-227-7006

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Final Report for AOARD Grant FA2386-14-1-4058

"**Large-scale Linear Optimization through Machine Learning**"

8 August 2016

Name of Principal Investigators (PI and Co-PIs): Jinwoo Shin
- E-mail address : jinwoos@kaist.ac.kr
- Institution : Korea Advanced Institute of Science and Technology
- Mailing Address : 335 Gwahangno, Yuseong-gu, Daejeon, Republic of Korea, 305701
- Phone : 82-42-350-7432

Period of Performance: 28 May 14 – 27 May 16

Abstract: The linear programming (LP) is one of the most popular necessary optimization tool used for data analytics as well as in various scientific fields. However, the current state-of-art algorithms suffer from scalability issues when processing Big Data. For example, the commercial optimization software IBM CPLEX cannot handle an LP with more than hundreds of thousands variables or constraints. Existing algorithms are fundamentally hard to scale because they are inevitably too complex to parallelize. To address the issue, we study the possibility of using the Belief Propagation (BP) algorithm as an LP solver. BP has shown remarkable performances on various machine learning tasks and it naturally lends itself to fast parallel implementations. Despite this, very little work has been done in this area. In particular, while it is generally believed that BP implicitly solves an optimization problem, it is not well understood under what conditions the solution to a BP converges to that of a corresponding LP formulation.

Our efforts consist of two main parts. First, we perform a theoretic study and establish the conditions in which BP can solve LP [1,2]. Although there has been several works studying the relation between BP and LP for certain instances, our work provides a generic condition unifying all prior works for generic LP. Second, utilizing our theoretical results, we develop a practical BP-based parallel algorithms for solving generic LPs, and it shows 71x speed up while sacrificing only 0.1% accuracy compared to the state-of-art exact algorithm [3, 4].

As a result of the study, the PIs have published two conference papers [1,3] and two follow-up journal papers [3,4] are under submission. We refer the readers to our published work [1,3] for details.

Introduction: The main goal of our research is to develop a distributed and parallel algorithm for large-scale linear optimization (or programming). Considering the popularity and importance of linear optimizations in various fields, the proposed method has great potentials applicable to various big data analytics. Our approach is based on the Belief Propagation (BP) algorithm, which has shown remarkable performances on various machine learning tasks and naturally lends itself to fast parallel implementations.

Our key contributions are summarized below:

1) We establish key theoretic foundations in the area of Belief Propagation. In particular, we show that BP converges to the solution of LP if some sufficient conditions are satisfied. Our

conditions not only cover various prior studies including maximum weight matching, min-cost network flow, shortest path, etc., but also discover new applications such as vertex cover and traveling salesman.

2) While the theoretic study provides understanding of the nature of BP, it falls short in slow convergence speed, oscillation and wrong convergence. To make BP-based algorithms more practical, we design a BP-based framework which uses BP as a 'weight transformer' to resolve the convergence issue of BP.

We refer the readers to our published work [1, 3] for details. The rest of the report contains a summary of our work appeared in UAI (Uncertainty in Artificial Intelligence) and IEEE Conference in Big Data [1,3] and follow up work [2,4] under submission to major journals.

Experiment: We first establish theoretical conditions when Belief Propagation (BP) can solve Linear Programming (LP), and second provide a practical distributed/parallel BP-based framework solving generic optimizations. We demonstrate the wide-applicability of our approach via popular combinatorial optimizations including maximum weight matching, shortest path, traveling salesman, cycle packing and vertex cover.

Results and Discussion: Our contribution consists of two parts: Study 1 [1,2] looks at the theoretical conditions that BP converges to the solution of LP. Our theoretical result unify almost all prior result about BP for combinatorial optimization. Furthermore, our conditions provide a guideline for designing distributed algorithm for combinatorial optimization problems. Study 2 [3,4] focuses on building an optimal framework based on the theory of Study 1 for boosting the practical performance of BP. Our framework is generic, thus, it can be easily extended to various optimization problems. We also compare the empirical performance of our framework to other heuristics and state of the art algorithms for several combinatorial optimization problems.

---------- Study 1 ----------

We first introduce the background for our contributions. A joint distribution of $\square$ (binary) variables $\square = [\square_{\square}] \in \{0,1\}^{\square}$ is called graphical model (GM) if it factorizes as follows: for $\square = [\square_{\square}] \in \{0,1\}^{\square}$,

$$\Pr[Z = z] \propto \prod_{i \in \{1,\ldots,n\}} \psi_i(z_i) \prod_{\alpha \in F} \psi_\alpha(z_\alpha),$$

where $\psi_{\square}$, $\square_{\square}$ are some non-negative functions so called factors; $\square$ is a collection of subsets

$$F = \{\alpha_1, \alpha_2, \ldots, \alpha_k\} \subset 2^{\{1,2,\ldots,n\}}$$

(each $\alpha_{\square}$ is a subset of $\{1, \cdots, \square\}$ with $|\square_{\square}| \geq 2$; $\square_{\square}$ is the projection of $\square$ onto dimensions included in α. Assignment $\square^*$ is called maximum-a-posteriori (MAP) assignment if $\square^*$maximizes the probability.

The following figure depicts the graphical relation between factors $\square$ and variables $\square$.

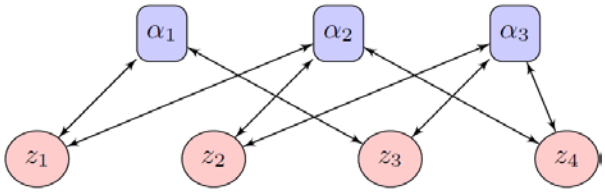

Figure 1: Factor graph for the graphical model with factors $\alpha_1 = \{1,3\}$, $\alpha_2 = \{1,2,4\}$, $\alpha_3 = \{2,3,4\}$

Now we introduce the algorithm, (max-product) BP, for approximating MAP assignment in a graphical model. BP is an iterative procedure; at each iteration t, there are four messages

$$\{m^t_{\alpha\to i}(c), m^t_{i\to\alpha}(c) : c \in \{0,1\}\}$$

between each variable x_i and every associated $\alpha \in \partial_i$, where $\partial_i := \{\alpha \in \partial : i \in \alpha\}$. Then, messages are updated as follows:

$$m^{t+1}_{\alpha\to i}(c) \;=\; \max_{z_\alpha : z_i = c} \psi_\alpha(z_\alpha) \prod_{j\in\alpha\setminus i} m^t_{j\to\alpha}(z_j) \tag{1}$$

$$m^{t+1}_{i\to\alpha}(c) \;=\; \psi_i(c) \prod_{\alpha'\in F_i\setminus\alpha} m^t_{\alpha'\to i}(c). \tag{2}$$

Finally, given messages, BP marginal beliefs are computed as follows:

$$b_i[z_i] \;=\; \prod_{\alpha\in F_i} m_{\alpha\to i}(z_i). \tag{3}$$

Then, BP outputs the approximated MAP assignment $z^{BP} = [z^{BP}_1, \ldots, z^{BP}_n]$ as

$$z^{BP}_i = \begin{cases} 1 & \text{if } b_i[1] > b_i[0] \\ ? & \text{if } b_i[1] = b_i[0] \\ 0 & \text{if } b_i[1] < b_i[0] \end{cases}.$$

Now, we are ready to introduce the main result of Study 1. Consider the following GM: for $x = [x_1, \ldots, x_n] \in \{0,1\}^n$ and $w = [w_1, \ldots, w_n] \in \mathbb{R}^n$,

$$\Pr[X = x] \;\propto\; \prod_i e^{-w_i x_i} \prod_{\alpha\in F} \psi_\alpha(x_\alpha), \tag{4}$$

where the factor function ψ_α for $\alpha \in \partial$ is defined as

$$\psi_\alpha(x_\alpha) = \begin{cases} 1 & \text{if } A_\alpha x_\alpha \geq b_\alpha,\; C_\alpha x_\alpha = d_\alpha \\ 0 & \text{otherwise} \end{cases},$$

for some matrices A_α, C_α and vectors b_α, d_α. Consider the Linear Programming (LP) corresponding the above GM:

$$\begin{aligned} \text{minimize} \quad & w \cdot x \\ \text{subject to} \quad & \psi_\alpha(x_\alpha) = 1, \quad \forall \alpha \in F \\ & x = [x_i] \in [0,1]^n. \end{aligned} \tag{5}$$

One can easily observe that the MAP assignments for GM corresponds to the (optimal) solution of the above LP if the LP has an integral solution $x^* \in \{0,1\}^n$. The following theorem is our main result of Study 1 which provide sufficient conditions so that BP can indeed find the LP solution

Theorem 1 *The max-product BP on GM (4) with arbitrary initial message converges to the solution of LP (5) if the following conditions hold:*

C1. LP (5) has a unique integral solution $x^* \in \{0,1\}^n$, i.e., it is tight.

C2. For every $i \in \{1,2,\ldots,n\}$, the number of factors associated with x_i is at most two, i.e., $|F_i| \leq 2$.

C3. For every factor ψ_α, every $x_\alpha \in \{0,1\}^{|\alpha|}$ with $\psi_\alpha(x_\alpha) = 1$, and every $i \in \alpha$ with $x_i \neq x_i^*$, there exists $\gamma \subset \alpha$ such that

$$|\{j \in \{i\} \cup \gamma : |F_j| = 2\}| \leq 2$$

$$\psi_\alpha(x'_\alpha) = 1, \qquad \text{where } x'_k = \begin{cases} x_k & \text{if } k \notin \{i\} \cup \gamma \\ x_k^* & \text{otherwise} \end{cases}.$$

$$\psi_\alpha(x''_\alpha) = 1, \qquad \text{where } x''_k = \begin{cases} x_k & \text{if } k \in \{i\} \cup \gamma \\ x_k^* & \text{otherwise} \end{cases}.$$

Theorem 1 can be applied to several combinatorial optimization problems including matching, network flow, shortest path, vertex cover, etc. See [1,2] for the detailed proof of Theorem 1 and its applications to various combinatorial optimizations including maximum weight matching, min-cost network flow, shortest path, vertex cover and traveling salesman.

---------- Study 2 ----------

Study 2 mainly focuses on providing a distributed generic BP-based combinatorial optimization solver which has high accuracy and low computational complexity. In summary, the key contributions of Study 2 are as follows:

1) Practical BP-based algorithm design: To the best of our knowledge, this paper is the first to propose a generic concept for designing BP-based algorithms that solve large-scale combinatorial optimization problems.
2) Parallel implementation: We also demonstrate that the algorithm is easily parallelizable. For the maximum weighted matching problem, this translates to 71x speed up while sacrificing only 0.1% accuracy compared to the state-of-art exact algorithm.
3) Extensive empirical evaluation: We evaluate our algorithms on three different combinatorial optimization problems on diverse synthetic and real-world data-sets. Our evaluation shows that the framework shows higher accuracy compared to other known heuristics.

Designing a BP-based algorithm for some problem is easy in general. However (a) it might diverge or converge very slowly, (b) even if it converges quickly, the BP decision might be not correct, and (c) even worse, BP might produce an infeasible solution, i.e., it does not satisfy the constraints of the problem.

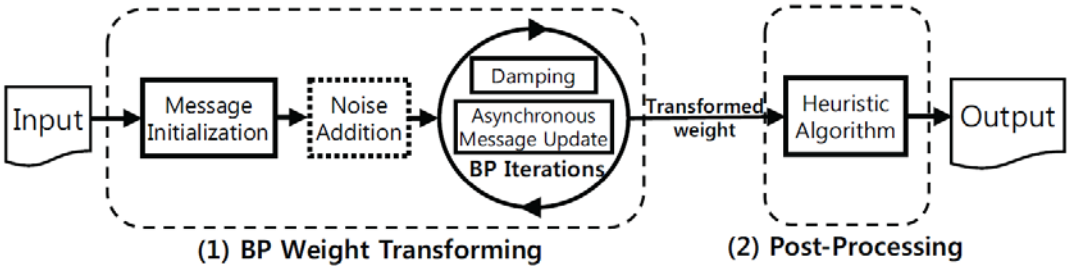

Figure 2: Overview of our generic BP-based framework

To address these issues, we propose a generic BP-based framework that provides highly accurate approximate solutions for combinatorial optimization problems. The framework has two steps, as shown in Figure 2. In the first phase, it runs a BP algorithm for a fixed number of iterations without waiting for convergence. Then, the second phase runs a known heuristic using BP beliefs instead of the original weights to output a feasible solution. Namely, the first and second phases are respectively designed for 'BP weight transforming' and 'post-processing'. Note that our evaluation mainly uses the maximum weight matching problem. The formal description of the maximum weight matching (MWM) problem is as follows: Given a graph $\square = (\square, \square)$ and edge weights $\square = [\square_{\square}] \in \square^{|\square|}$, it finds a set of edges such that each vertex is connected to at most one edge in the set and the sum of edge weights in the set is maximized. The problem is formulated as the following IP (Integer Programming):

$$\begin{aligned} &\text{maximize} \quad w \cdot x \\ &\text{s.t.} \quad \sum_{e \in \delta(v)} x_e \leq 1, \ \forall v \in V, \quad x = [x_e] \in \{0,1\}^{|E|}, \end{aligned}$$

where $\delta(\square)$ is the set of edges incident to vertex $\square \in \square$. In the following paragraphs, we describe the two phases in more detail in reverse order.

We first describe the post-processing phase. As we mentioned, one of the main issue of a BP-based algorithm is that the decision on BP beliefs might give an infeasible solution. To resolve the issue, we use post-processing by utilizing existing heuristics to the given problem that find a feasible solution. Applying post-processing ensures that the solution is at least feasible. In addition, our key idea is to replace the original weights by the logarithm of BP beliefs, i.e. function of (3). After this, we apply known heuristics using the logarithm of BP beliefs to achieve higher accuracy.

To confirm the effectiveness of the proposed post-processing mechanism, we compare it with the following two alternative post-processing schemes for the maximum weight matching problem that remove edges to enforce matching after BP processing in a naive manner:

- Random: If there exists a vertex $\square$ such that more than one neighboring edges are selected on the BP decision, randomly select one edge and remove other edges.
- Weight: If there exists a vertex $\square$ such that more than one neighboring edges are selected on the BP decision, remove edges of smaller weight.

Figure 3 compares the approximation ratio obtained using BP-belief-based post-processing versus the naive post-processing heuristics (random and weight). It shows that the proposed BP-belief-based post-processing outperforms the rest. Note, the results in Figure 3 were obtained by first applying BP message passing for weight transformation. Next, we explain how this is done in our framework.

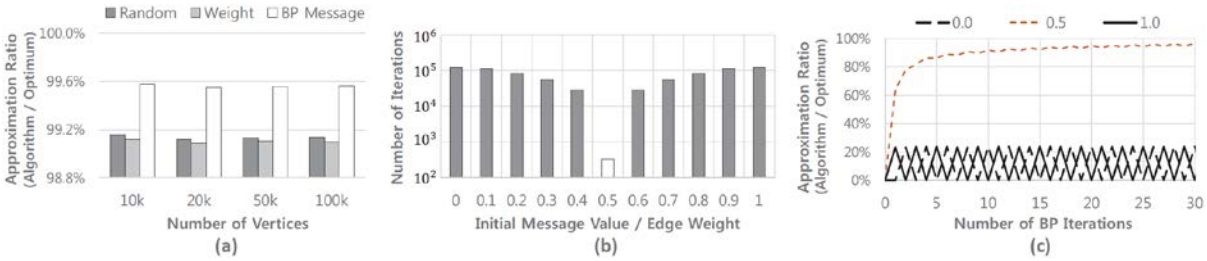

Figure 3: a) Average approximation ratio for different post-processing schemes. We use a local greedy algorithm as a post-processing based on original weights and BP messages (i.e., beliefs). The 'Random selection' post-processing is also compared. b) Effects of initial messages on the convergence of BP. We set $\square_{\square\square} := \square\square\square \frac{\square_{\square\to(\square,\square)}{}^{0}(0)}{\square_{\square\to(\square,\square)}{}^{0}(1)} = \square\square_{\square\square}$ where x-axis represent the value of $\square$. c) Approximation ratio for different initial messages $\square_{\square\square} = 0, \square_{\square\square}/2, \square_{\square\square}$.

Now, we describe the BP weight transforming phase. To improve the approximation quality and solve the convergence issues, we use three modifications to the standard BP algorithm: (1) careful initialization on messages, (2) noise addition and (3) hybrid damping on message updates.

Message Initialization. The standard message initialization is $\square_{\square\to\square}{}^{0} = \square_{\square\to\square}{}^{0} = 1$ for the maximum weight matching problem. However, the convergence rate of BP depends on the initialized messages. As reported in Figure 4, we try different initializations by varying the log ratio $\square_{\square\square} := \square\square\square \frac{\square_{\square\to(\square,\square)}{}^{0}(0)}{\square_{\square\to(\square,\square)}{}^{0}(1)} = \square\square_{\square\square}$ for $0 \le \square \le 1$, where the case $\square = 0.5$ shows the fastest convergence. The choice $\square_{\square\square} = 0.5\square_{\square\square}$ alleviates the fluctuation behavior of BP and boosts up its convergence speed. We remind that, under our framework, BP runs only for a fixed number of iterations since it might converge too slowly, even with the initialization $\square_{\square\square} = 0.5\square_{\square\square}$ for practical purposes. With fixed number of iterations, careful initialization becomes even more critical as experimental results in Figure 3(c) and Figure 4 suggest. For example, if one runs 5000 iterations of BP, they show that the standard initialization achieves at most 30% approximation ratio, while the proposed method achieves 99%. Moreover, one can also observe that the advantage of more BP updates diminishes as the number of iterations

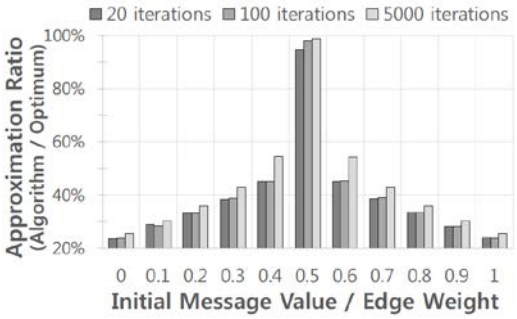

Figure 4: Effects of initial messages on the number of BP iterations. We set $\square_{\square\square} = \square\square_{\square\square}$ for a value c of x-axis.

Noise Addition. The BP algorithm often oscillates when the MAP solution is not unique. To address this issue, we transform the original problem to one that has a unique solution with high probability by adding small noises to the weights. We apply this to all cases. Here, one has to be careful in deciding the range $\square$ of noises. If $\square$ is too large, the quality of BP solution deteriorates because the optimal solution might have changed from the original problem. On the other hand, if $\square$ is too small

compared to we, BP converges very slowly. To achieve a balance, we choose the range r of noise re as 10% of the minimum distance among weights. We find that this results in over 99.8% approximation ratio even when the solution is not unique, which has little difference with that of unique solution as shown in Table I.

# vertices (# edges)	Approximation Ratio		Difference
	Multiple optima	Unique optimum	
1k (50k)	99.88 %	99.90 %	-0.02 %
5k (250k)	99.86 %	99.85 %	+0.01 %
10k (500k)	99.85 %	99.84 %	+0.01 %
20k (1M)	99.84 %	99.83 %	+0.01 %

Table I: Approximation ratio of BP for MWM with multiple optima and a unique optimum. We introduce a small noise to the edge weights and set the initial message by $\square_{\square\square} = \square_{\square\square}/2$.

Hybrid Damping. To boost up the convergence speed of BP updates further, we use a specific damping strategy to alleviate message oscillation. We update messages to be the average of old and new messages as follows:

$$a_{i\to j}^{t+1} \leftarrow \max_{k\in\delta(i)\setminus\{j\}} \left\{\max\left\{w_{ik} - a_{k\to i}^{t}, 0\right\}\right\}$$
$$a_{i\to j}^{t+1} \leftarrow (a_{i\to j}^{t} + a_{i\to j}^{t+1})/2.$$

We note that the damping strategy provides a similar effect as our proposed initialization $\square_{\square\square} = \square_{\square\square}/2$. Hence, if one uses both, the effect of one might be degraded due to the other. Due to this, we first run the half of BP iterations without damping (this is for keeping the effect of the proposed initialization) and perform the last half of BP iterations with damping. As reported in Table II, this hybrid approach outperforms other alternatives, including (a) no use of damping, (b) using damping in every iteration, and (c) damping in the first half of BP iterations and no-damping in the last half.

# vertices (# edges)	Approximation Ratio			
	no-damp(100)	damp(100)	no-damp(50) +damp(50)	damp(50) +no-damp(50)
10k (500k)	99.58 %	99.69 %	**99.83 %**	99.56 %
20k (1M)	99.55 %	99.68 %	**99.82 %**	99.56 %
50k (2.5M)	99.56 %	99.69 %	**99.83 %**	99.57 %
100k (5M)	99.56 %	99.69 %	**99.83 %**	99.57 %

Table II: Approximation ratio of BP without damping, BP with damping, BP with damping only for first 50 iterations, and BP with damping for last 50 iterations. We introduce a small noise to the edge weights and set the initial message by $\square_{\square\square} = \square_{\square\square}/2$.

Now we describe the implementation, mostly parallelization, of our framework. First, we introduce asynchronous message update that enables efficient parallelization of BP message passing. Second, we illustrate the issues in parallelizing post-processing. Finally, we describe the parallel implementations of our algorithm and their benefits.

Asynchronous Message Update. For parallelization, we first divide the graph by partitioning the vertices, and assign each partition to a single thread. However, if we naively parallelize the process using multiple threads, frequent synchronization may incur large overhead. Thus, we apply asynchronous message update where each vertex updates the message value right after new message value is calculated and eliminate synchronization point between iterations. This makes the process faster because of the reduced synchronization points. Figure 5 shows that performance improvement

(speed up in running time) of asynchronous update over synchronous is up to 237% in our example graph for the maximum weight matching problem with 16 threads

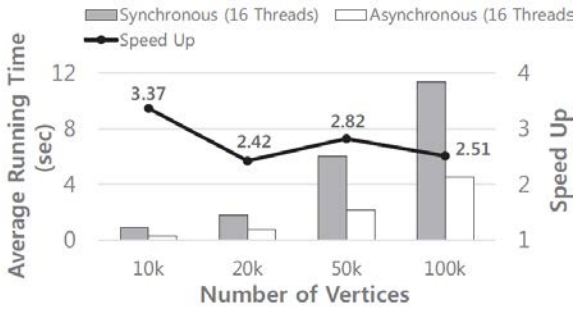

Figure 5: Average running time of our BP-based algorithm with synchronous message update and asynchronous message update.

Local Post-Processing. The second phase of our algorithm runs existing heuristics for post-processing to enforce the feasibility of BP decisions. While the framework works with any heuristics-based post-processing methods, for the entire process to be parallel, it is important that the post-processing step is also parallel. An important criterion for efficient parallelization is locality of computation; if the post-processing heuristics can compute the result locally without requiring global knowledge, they can be easily parallelized. Moreover, if they do not require synchronization, the running time can be further reduced.

Parallel Implementation. The BP algorithm is easy to parallelize because of its message passing nature. To demonstrate this, we parallelize our BP-based framework using three platforms: GraphChi, OpenMP and pthread.

Now we show the empirical performance of our framework using three popular combinatorial optimization problems: maximum weight matching, minimum weight vertex cover (MWVC) and maximum weight independent set problem (MWIS). We already introduced the IP formulation of the maximum weight matching (MWM), where those of the minimum weight vertex cover (MWVC) and maximum weight independent set problem (MWIS) are as follows:

$$\begin{aligned}
\textbf{MWVC:}\quad & \text{minimize} \quad w \cdot x \\
& \text{subject to} \quad x_u + x_v \geq 1,\ \forall e = (u,v) \in E \\
& \qquad x = [x_v] \in \{0,1\}^{|V|} \\
\textbf{MWIS:}\quad & \text{maximize} \quad w \cdot x \\
& \text{subject to} \quad x_u + x_v \leq 1,\ \forall e = (u,v) \in E \\
& \qquad x = [x_e] \in \{0,1\}^{|V|}.
\end{aligned}$$

Experimental Setup. In our experiments, both real-world and synthetic datasets are used for evaluation. For MWM, we used data-sets from the university of Florida sparse matrix collection. For larger scale synthetic evaluation, we generate Erdős-Rényi random graphs (up to 50 million vertices with 2.5 billion edges) with average vertex degree of 100 with edge weights drawn independently from the uniform random distribution over the interval [0, 1]. For MWVC and MWIS, we use the frb-series from BHOSLIB, where it also contains the optimal solutions. We note that we perform no experiment using synthetic data-sets for MWVC and MWIS since they are NP-hard problems, i.e., impossible to compute the optimal solutions. On the other hand, for MWM the Edmonds' Blossom algorithm can compute the optimal solution in polynomial time. All experiments in this section are

conducted on a machine with Intel Xeon(R) CPU E5-2690 @ 2.90GHz with 8 cores and 8 hyperthreads with 128GB of memory, unless otherwise noted.

Approximation Ratio. We now demonstrate our BP-based approximation algorithm produces highly accurate results. In particular, we show that our BP-based algorithms outperform well-known heuristics for MWVC, MWIS and closely approximate exact solutions for MWM for all cases we evaluate.

- For MWM, we compare the approximation qualities of serial, synchronous BP and parallel, asynchronous implementation on both synthetic and real-world data-sets, where we compute the optimal solution using the Blossom algorithm to measure the approximation ratios. Table III summarize our experimental results for MWM for the synthetic data-sets and the Florida data. Our BP-based algorithm achieves 99% to 99.9% approximation ratios.

		Approximation Ratio	
		Serial BP (1 thread)	Parallel BP (16 threads)
Synthetic # vertices (# edges)	500k (25M)	99.93 %	99.90 %
	1M (50M)	99.93 %	99.90 %
	2M (100M)	99.94 %	99.91 %
	5M (250M)	99.93 %	99.90 %
Real-world Data-set (Florida)	apache1	100.0 %	100.0 %
	apache2	100.0 %	100.0 %
	ecology2	100.0 %	100.0 %
	G3_circuit	99.95 %	99.95 %
	bone010	99.11 %	99.12 %

Table III: MWM: Approximation ratio of our BP-based algorithm on synthetic and sparse matrix collection data-sets

- For MWVC, we use two post-processing procedures: greedy and 2-approximation algorithm. For the local greedy algorithm, we choose a random edge and add one of its adjacent vertices with a smaller weight until all edges are covered. We compare the approximation qualities of our BP-based algorithm compared to the cases when one uses only the greedy algorithm and the 2-approximation algorithm. Figure 6 shows the experimental results for the two post-processing heuristics. The results show that our BP-based weight transformation enhances the approximation quality of known approximation heuristics by up to 43%.

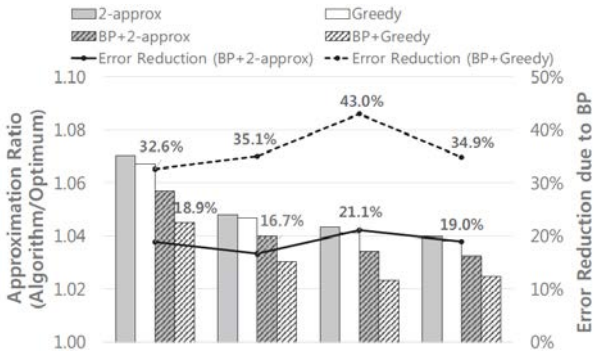

Figure 6: MWVC: Average approximation ratio of our BP-based algorithm, the 2-approximation algorithm and the greedy algorithm on frb-series data-sets.

- For MWIS, the experiment was performed on frb-series data-sets. We use a greedy algorithm as the post-processing procedure, which selects vertices in the order of higher weights until no

vertex can be selected without violating the independent set constraint. We compare the approximation qualities of our BP-based algorithm and the standard greedy algorithm. Figure 7 shows that our BP-based framework enhances the approximation ratio of the solution by 2% to 23%.

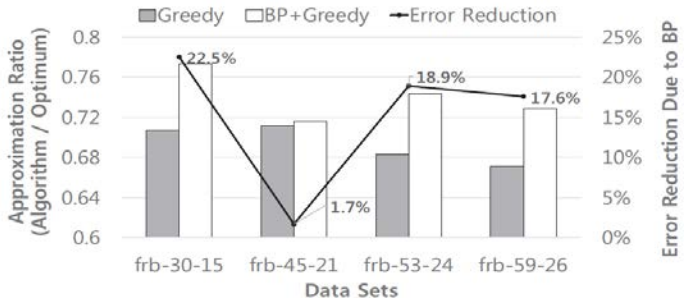

Figure 7: MWIS: Average approximation ratio of our BP-based algorithm and the greedy algorithm on frb-series data-sets.

Parallelization Speed-up. Figure 8 compares the running time of the Blossom algorithm and our BP-based algorithm with 1 single core and 16 cores. With five million vertices, our asynchronous parallel implementation is eight times faster than the synchronous serial implementation, while still retaining 99.9% approximation ratio as reported in Table III. To demonstrate the overall benefit in context, we compare its running time with that of the current fastest implementation of the Blossom algorithm due to Kolmogorov. Here, we note that the Blossom algorithm is inherently not easy to parallelize. For parallel implementation, we report results for our pthread implementation, but the OpenMP implementation also show comparable performance. For 20 million vertices (one billion edges), it shows that the running time of our algorithm can be accelerated by up to 71 times than the Blossom algorithm, while sacrificing 0.1% of accuracy. The running time gap is expected be more significant for larger graphs since the running times of our algorithm and the Blossom algorithm are linear and cubic with respect to the number of vertices, respectively.

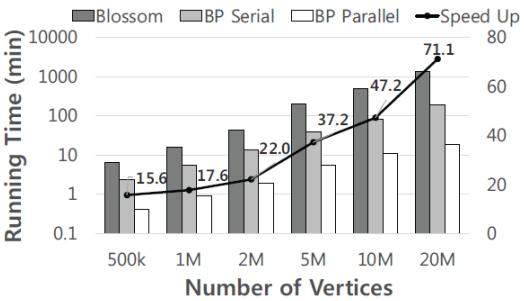

Figure 8: MWM: Running time of Blossom algorithm and our BP-based algorithms.

Large-scale Optimization. Our algorithm can also handle large-scale instances because it is based on GMs that inherently lend itself to parallel and distributed implementations. To demonstrate this, we create a large-scale instance containing up to 50 million vertices and 2.5 billion edges. We experiment our algorithm using GraphChi on a single consumer level machine with i7 CPU and 24GB of memory. Figure 9 shows the running time and memory usage of our algorithm for MWM and MWVC on large data-sets.

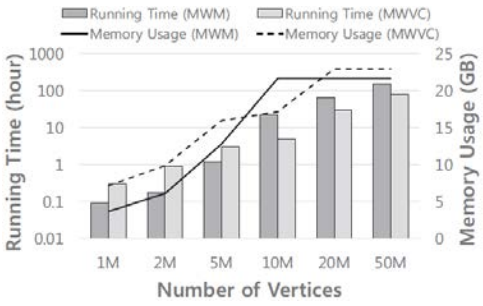

Figure 9: MWM and MWVC: Running time and memory usage of GraphChi-based implementation on large-scale graphs.

List of Publications and Significant Collaborations that resulted from your AOARD supported project: In standard format showing authors, title, journal, issue, pages, and date, for each category list the following:

a) papers published in peer-reviewed journals,

b) papers published in peer-reviewed conference proceedings,

[1] Sejun Park and Jinwoo Shin, Max-Product Belief Propagation for Linear Programming: Applications to Combinatorial Optimization, Conference on Uncertainty in Artificial Intelligence (UAI) 2015

[3] Inho Cho, Soya Park, Sejun Park, Dongsu Han and Jinwoo Shin, Practical Message-passing Framework for Large-scale Combinatorial Optimization, IEEE International Conference on Big Data (IEEE BigData) 2015

c) papers published in non-peer-reviewed journals and conference proceedings,

d) conference presentations without papers,

e) manuscripts submitted but not yet published, and

[2] Sejun Park and Jinwoo Shin, Convergence and Correctness of Max-Product Belief Propagation for Linear Programming, under the second round of revision in SIAM J. Discrete Math (SIDMA)

[4] Inho Cho, Soya Park, Sejun Park, Dongsu Han and Jinwoo Shin, Large-scale Combinatorial Optimization via Belief Propagation: Practical Perspective, submitted to IEEE Transaction on Parallel and Distributed Systems (TPDS)

f) provide a list any interactions with industry or with Air Force Research Laboratory scientists or significant collaborations that resulted from this work.

Attachments: Publications a), b) and c) listed above if possible.

Max-Product Belief Propagation for Linear Programming: Applications to Combinatorial Optimization

Sejun Park
Department of Electrical Engineering
Korea Advanced Institute of Science and Technology
sejun.park@kaist.ac.kr

Jinwoo Shin
Department of Electrical Engineering
Korea Advanced Institute of Science and Technology
jinwoos@kaist.ac.kr

Abstract

Max-product belief propagation (BP) is a popular message-passing algorithm for computing a maximum-a-posteriori (MAP) assignment in a joint distribution represented by a graphical model (GM). It has been shown that BP can solve a few classes of Linear Programming (LP) formulations to combinatorial optimization problems including maximum weight matching and shortest path, i.e., BP can be a distributed solver for certain LPs. However, those LPs and corresponding BP analysis are very sensitive to underlying problem setups, and it has been not clear what extent these results can be generalized to. In this paper, we obtain a generic criteria that BP converges to the optimal solution of given LP, and show that it is satisfied in LP formulations associated to many classical combinatorial optimization problems including maximum weight perfect matching, shortest path, traveling salesman, cycle packing and vertex cover. More importantly, our criteria can guide the BP design to compute fractional LP solutions, while most prior results focus on integral ones. Our results provide new tools on BP analysis and new directions on efficient solvers for large-scale LPs.

1 INTRODUCTION

Graphical model (GM) has been one of powerful paradigms for succinct representations of joint probability distributions in variety of scientific fields (Yedidia et al., 2005; Richardson and Urbanke, 2008; Mezard and Montanari, 2009; Wainwright and Jordan, 2008). GM represents a joint distribution of some random vector to a graph structured model where each vertex corresponds to a random variable and each edge captures to a conditional dependency between random variables. In many applications involving GMs, finding maximum-a-posteriori (MAP) assignment in GM is an important inference task, which is known to be computationally intractable (i.e., NP-hard) in general (Chandrasekaran et al., 2008). Max-product belief propagation (BP) is the most popular heuristic for approximating a MAP assignment of given GM, where its performance has been not well understood in loopy GMs. Nevertheless, BP often shows remarkable performances even on loopy GM. Distributed implementation, associated ease of programming and strong parallelization potential are the main reasons for the growing popularity of the BP algorithm. For example, several software architectures for implementing parallel BPs were recently proposed (Low et al., 2010; Gonzalez et al., 2010; Ma et al., 2012) by different research groups in machine learning communities.

In the past years, there have been made extensive research efforts to understand BP performances on loopy GMs behind its empirical success. Several characterizations of the max-product BP fixed points have been proposed (Weiss and Freeman, 2001; Vinyals et al., 2010), whereas they do not guarantee the BP convergence in general. It has also been studied about the BP convergence to the correct answer, in particular, under a few classes of loopy GM formulations of combinatorial optimization problems: matching (Bayati et al., 2005; Sanghavi et al., 2011; Huang and Jebara, 2007; Salez and Shah, 2009), perfect matching (Bayati et al., 2011), matching with odd cycles (Shin et al., 2013) and shortest path (Ruozzi and Tatikonda, 2008). The important common feature of these instances is that BP converges to a correct MAP assignment if the Linear Programming (LP) relaxation of the MAP inference problem is tight, i.e., it has no integrality gap. In other words, BP can be used an efficient distributed solver for those LPs, and is presumably of better choice than classical centralized LP solvers such as simplex methods (Dantzig, 1998), interior point methods (Thapa, 2003) and ellipsoid methods (Khachiyan, 1980) for large-scale inputs. However, these theoretical results on BP are very sensitive to underlying structural properties depending on specific problems and it is not clear what extent they can be generalized to, e.g., the BP analysis for matching problems (Bayati et al., 2005; Sanghavi et al., 2011; Huang and Jebara, 2007; Salez and Shah, 2009) are not extended to even for perfect matching

ones (Bayati et al., 2011). In this paper, we overcome such technical difficulties for enhancing the power of BP as a LP solver.

Contribution. We establish a generic criteria for GM formulations of given LP so that BP converges to the optimal LP solution. By product, it also provides a sufficient condition for a unique BP fixed point. As one can naturally expect given prior results, one of our conditions requires the LP tightness. Our main contribution is finding other sufficient generic conditions so that BP converges to the correct MAP assignment of GM. First of all, our generic criteria can rediscover all prior BP results on this line, including matching (Bayati et al., 2005; Sanghavi et al., 2011; Huang and Jebara, 2007), perfect matching (Bayati et al., 2011), matching with odd cycles (Shin et al., 2013) and shortest path (Ruozzi and Tatikonda, 2008), i.e., we provide a unified framework on establishing the convergence and correctness of BPs in relation to associated LPs. Furthermore, we provide new instances under our framework: we show that BP can solve LP formulations associated to other popular combinatorial optimizations including perfect matching with odd cycles, traveling salesman, cycle packing and vertex cover, which are not known in the literature. While most prior known BP results on this line focused on the case when the associated LP has an integral solution, the proposed criteria naturally guides the BP design to compute fractional LP solutions as well (see Section 4.2 and Section 4.4 for details).

Our proof technique is built upon on that of Sanghavi et al. (2011) where the authors construct an alternating path in the computational tree induced by BP to analyze its performance for the maximum weight matching problem. Such a trick needs specialized case studies depending on the associated LP when the path reaches a leaf of the tree, and this is one of main reasons why it is not easy to generalize to other problems beyond matching. The main technical contribution of this paper is providing a way to avoid the issue in the BP analysis via carefully analyzing associated LP polytopes.

The main appeals of our results are providing not only tools on BP analysis, but also guidelines on BP design for its high performance, i.e., one can carefully design a BP given LP so that it satisfies the proposed criteria. We run such a BP for solving the famous traveling saleman problem (TSP), and our experiments show that BP outperforms other popular heuristics (see Section 5). Our results provide not only new tools on BP analysis and design, but also new directions on efficient distributed (and parallel) solvers for large-scale LPs and combinatorial optimization problems.

Organization. In Section 2, we introduce necessary backgrounds for the BP algorithm. In Section 3, we provide the main result of the paper, and several concrete applications to popular combinatorial optimizations are described in Section 4. In Section 5, we show empirical performances of BP algorithms for solving TSP.

2 PRELIMINARIES

2.1 GRAPHICAL MODEL

A joint distribution of n (binary) random variables $Z = [Z_i] \in \{0,1\}^n$ is called a Graphical Model (GM) if it factorizes as follows: for $z = [z_i] \in \{0,1\}^n$,

$$\Pr[Z = z] \propto \prod_{i \in \{1,\dots,n\}} \psi_i(z_i) \prod_{\alpha \in F} \psi_\alpha(z_\alpha),$$

where $\{\psi_i, \psi_\alpha\}$ are (given) non-negative functions, so-called factors; F is a collection of subsets

$$F = \{\alpha_1, \alpha_2, ..., \alpha_k\} \subset 2^{\{1,2,\dots,n\}}$$

(each α_j is a subset of $\{1, 2, \dots, n\}$ with $|\alpha_j| \geq 2$); z_α is the projection of z onto dimensions included in α.[1] In particular, ψ_i is called a variable factor. Figure 1 depicts the the graphical relation between factors F and variables z.

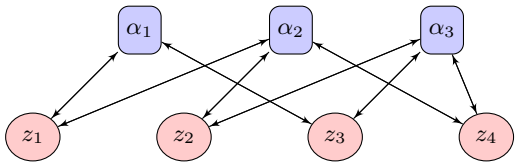

Figure 1: Factor graph for the graphical model $\Pr[z] \propto \psi_{\alpha_1}(z_1, z_3)\psi_{\alpha_2}(z_1, z_2, z_4)\psi_{\alpha_3}(z_2, z_3, z_4)$, i.e., $F = \{\alpha_1, \alpha_2, \alpha_3\}$ and $n = 4$. Each α_j selects a subset of z. For example, α_1 selects $\{z_1, z_3\}$.

Assignment z^* is called a maximum-a-posteriori (MAP) assignment if $z^* = \arg\max_{z \in \{0,1\}^n} \Pr[z]$. This means that computing a MAP assignment requires us to compare $\Pr[z]$ for all possible z, which is typically computationally intractable (i.e., NP-hard) unless the induced bipartite graph of factors F and variables z, so-called factor graph, has a bounded treewidth (Chandrasekaran et al., 2008).

2.2 MAX-PRODUCT BELIEF PROPAGATION

The (max-product) BP algorithm is a popular heuristic for approximating the MAP assignment in GM. BP is implemented iteratively; at each iteration t, BP maintains four messages $\{m^t_{\alpha \to i}(c), m^t_{i \to \alpha}(c) : c \in \{0,1\}\}$ between every variable z_i and every associated $\alpha \in F_i$, where $F_i := \{\alpha \in F : i \in \alpha\}$; that is, F_i is a subset of F such that all α in F_i are associated with z_i. The messages

[1] For example, if $z = [0,1,0]$ and $\alpha = \{1,3\}$, then $z_\alpha = [0,0]$.

are updated as follows:

$$m_{\alpha\to i}^{t+1}(c) = \max_{z_\alpha : z_i = c} \psi_\alpha(z_\alpha) \prod_{j\in\alpha\setminus i} m_{j\to\alpha}^t(z_j) \quad (1)$$

$$m_{i\to\alpha}^{t+1}(c) = \psi_i(c) \prod_{\alpha'\in F_i\setminus\alpha} m_{\alpha'\to i}^t(c). \quad (2)$$

Where each z_i only sends messages to F_i; that is, z_i sends messages to α_j only if α_j selects/includes i. The outer-term in the message computation (1) is maximized over all possible $z_\alpha \in \{0,1\}^{|\alpha|}$ with $z_i = c$. The inner-term is a product that only depends on the variables z_j (excluding z_i) that are connected to α. The message-update (2) from variable z_i to factor ψ_α is a product containing all messages received by z_i in the previous iteration, except for the message sent by ψ_α itself.

One can reduce the complexity by combining (1) and (2) as:

$$m_{i\to\alpha}^{t+1}(c) = \psi_i(c) \prod_{\alpha'\in F_i\setminus\alpha} \max_{z_{\alpha'} : z_i = c} \psi_{\alpha'}(z_{\alpha'}) \times \prod_{j\in\alpha'\setminus i} m_{j\to\alpha'}^t(z_j).$$

The BP fixed-point of messages is defined as $m^{t+1} = m^t$ under the above updating rule. Given a set of messages $\{m_{i\to\alpha}(c), m_{\alpha\to i}(c) : c \in \{0,1\}\}$, the so-called BP marginal beliefs are computed as follows:

$$b_i[z_i] \quad = \psi_i(z_i) \textstyle\prod_{\alpha\in F_i} m_{\alpha\to i}(z_i). \quad (3)$$

This BP algorithm outputs $z^{BP} = [z_i^{BP}]$ where

$$z_i^{BP} = \begin{cases} 1 & \text{if } b_i[1] > b_i[0] \\ ? & \text{if } b_i[1] = b_i[0] \\ 0 & \text{if } b_i[1] < b_i[0] \end{cases}.$$

It is known that z^{BP} converges to a MAP assignment after a sufficient number of iterations, if the factor graph is a tree and the MAP assignment is unique. However, if the graph contains cycles, the BP algorithm is not guaranteed to converge a MAP assignment in general.

3 CONVERGENCE AND CORRECTNESS OF BELIEF PROPAGATION

In this section, we provide the main result of this paper: a convergence and correctness criteria of BP. Consider the following GM: for $x = [x_i] \in \{0,1\}^n$ and $w = [w_i] \in \mathbb{R}^n$,

$$\Pr[X = x] \propto \prod_i e^{-w_i x_i} \prod_{\alpha\in F} \psi_\alpha(x_\alpha), \quad (4)$$

where F is the set of non-variable factors and the factor function ψ_α for $\alpha \in F$ is defined as

$$\psi_\alpha(x_\alpha) = \begin{cases} 1 & \text{if } A_\alpha x_\alpha \geq b_\alpha,\ C_\alpha x_\alpha = d_\alpha \\ 0 & \text{otherwise} \end{cases},$$

for some matrices A_α, C_α and vectors b_α, d_α. Now we consider the Linear Programming (LP) corresponding the above GM:

$$\begin{aligned} \text{minimize} \quad & w \cdot x \\ \text{subject to} \quad & \psi_\alpha(x_\alpha) = 1, \quad \forall \alpha \in F \\ & x = [x_i] \in [0,1]^n. \end{aligned} \quad (5)$$

One can easily observe that the MAP assignments for GM (4) corresponds to the (optimal) solution of LP (5) if the LP has an integral solution $x^* \in \{0,1\}^n$. As stated in the following theorem, we establish other sufficient conditions so that the max-product BP can indeed find the LP solution.

Theorem 1 *The max-product BP on GM (4) with arbitrary initial message converges to the solution of LP (5) if the following conditions hold:*

C1. LP (5) has a unique integral solution $x^* \in \{0,1\}^n$, i.e., it is tight.

C2. For every $i \in \{1,2,\ldots,n\}$, the number of factors associated with x_i is at most two, i.e., $|F_i| \leq 2$.

C3. For every factor ψ_α, every $x_\alpha \in \{0,1\}^{|\alpha|}$ with $\psi_\alpha(x_\alpha) = 1$, and every $i \in \alpha$ with $x_i \neq x_i^*$, there exists $\gamma \subset \alpha$ such that

$$|\{j \in \{i\} \cup \gamma : |F_j| = 2\}| \leq 2$$

$$\psi_\alpha(x'_\alpha) = 1, \qquad \text{where } x'_k = \begin{cases} x_k & \text{if } k \notin \{i\}\cup\gamma \\ x_k^* & \text{otherwise} \end{cases}.$$

$$\psi_\alpha(x''_\alpha) = 1, \qquad \text{where } x''_k = \begin{cases} x_k & \text{if } k \in \{i\}\cup\gamma \\ x_k^* & \text{otherwise} \end{cases}.$$

Since Theorem 1 holds for arbitrary initial messages, the conditions *C1, C2, C3* also provides the uniqueness of BP fixed-points in term of marginal beliefs, as follows.

Corollary 2 *The BP fixed-points of GM (4) have the same marginal beliefs if conditions C1, C2, C3 hold.*

The conditions *C2, C3* are typically easy to check given GM (4) and the uniqueness in *C1* can be easily guaranteed via adding random noises, where we provide several concrete examples in Section 4. On the other hand, the integral property in *C1* requires to analyze LP (5), where it has been extensively studied in the field of combinatorial optimization (Schrijver, 2003). Nevertheless, Theorem 1 provides important guidelines to design BP algorithms, irrespectively of the LP analysis. For example, in Section 5, we report empirical performances of BP following the above guideline for solving the traveling salesman problem, without relying on whether the corresponding LP has an integral solution or not.

3.1 PROOF OF THEOREM 1

To begin with, we define some necessary notation. We let $\mathcal{P}$ denote the polytope of feasible solutions of LP (5):

$$\mathcal{P} := \{x \in [0,1]^n \,:\, \psi_\alpha(x_\alpha) = 1,\ \forall \alpha \in F\}.$$

Similarly, $\mathcal{P}_\alpha$ is defined as

$$\mathcal{P}_\alpha := \left\{x \in [0,1]^{|\alpha|} \,:\, \psi_\alpha(x_\alpha) = 1\right\}.$$

We first state the following key technical lemma.

Lemma 3 *There exist universal constants $K, \eta > 0$ for LP (5) such that if $z \in [0,1]^n$ and $0 < \varepsilon < \eta$ satisfy the followings:*

1. *There exist at most two violated factors for z, i.e., $|\{\alpha \in F \,:\, z_\alpha \notin \mathcal{P}_\alpha\}| \le 2$.*
2. *For each violated factor α, there exist $i \in \alpha$ such that $z^\dagger_\alpha \in \mathcal{P}_\alpha$, where $z^\dagger = z + \varepsilon e_i$ or $z^\dagger = z - \varepsilon e_i$ and $e_i \in \{0,1\}^n$ is the unit vector whose i-th coordinate is 1,*

then there exists $z^\ddagger \in \mathcal{P}$ such that $\|z - z^\ddagger\|_1 \le \varepsilon K$.

The proof of Lemma 3 is presented in Section 3.2. Now, from Condition *C1*, it follows that there exists $\rho > 0$ such that

$$\rho := \inf_{x \in \mathcal{P} \setminus x^*} \frac{w \cdot x - w \cdot x^*}{\|x - x^*\|_1} > 0. \tag{6}$$

We let $\widehat{x}^t \in \{0,1,?\}^n$ denote the BP estimate at the t-th iteration for the MAP computation. We will show that under Conditions *C1*-*C3*,

$$\widehat{x}^t = x^*, \qquad \text{for } t > \left(\frac{w_{\max}}{\rho} + 1\right) K,$$

where $w_{\max} = \max_j |w_j|$ and K is the universal constant in Lemma 3. Suppose the above statement is false, i.e., there exists $i \in \{1,2,\ldots,n\}$ such that $\widehat{x}^t_i \neq x^*_i$ for $t > \left(\frac{w_{\max}}{\rho} + 1\right) K$. Under the assumption, we will reach a contradiction.

Now we construct a tree-structured GM $T_i(t)$, popularly known as the computational tree (Weiss and Freeman, 2001), as follows:

1. Add $y_i \in \{0,1\}$ as the root variable with variable factor function $e^{-w_i y_i}$.
2. For each leaf variable y_j and for each $\alpha \in F_j$ and ψ_α is not associated with y_j in the current tree-structured GM, add a factor function ψ_α as a child of y_j.
3. For each leaf factor ψ_α and for each variable y_k such that $k \in \alpha$ and y_k is not associated with ψ_α in the current tree-structured GM, add a variable y_k as a child of ψ_α with variable factor function $e^{-w_k y_k}$.
4. Repeat Step 2, 3 t times.

Suppose the initial messages of BP are set by 1, i.e., $m_{j\to\alpha}(\cdot)^0 = 1$. Then, if $x^*_i \neq \widehat{x}^t_i$, it is known (Weiss, 1997) that there exists a MAP configuration y^{MAP} on $T_i(t)$ with $y^{MAP}_i \neq x^*_i$ at the root variable. For other initial messages, one can guarantee the same property under changing weights of leaf variables of the tree-structured GM. Specifically, for a leaf variable k with $|F_k = \{\alpha_1, \alpha_2\}| = 2$ and α_1 being its parent factor in $T_i(t)$, we reset its variable factor by $e^{-w'_k y_k}$, where

$$w'_k = w_k - \log \frac{\max_{z_{\alpha_2} : z_k = 1} \psi_{\alpha_2}(z_{\alpha_2}) \Pi_{j \in \alpha_2 \setminus k} m^0_{j\to\alpha_2}(z_j)}{\max_{z_{\alpha_2} : z_k = 0} \psi_{\alpha_2}(z_{\alpha_2}) \Pi_{j \in \alpha_2 \setminus k} m^0_{j\to\alpha_2}(z_j)}. \tag{7}$$

This is the reason why our proof of Theorem 1 goes through for arbitrary initial messages. For notational convenience, we present the proof for the standard initial message of $m^0_{j\to\alpha}(\cdot) = 1$, where it can be naturally generalized to other initial messages using (7).

Now we construct a new valid assignment y^{NEW} on the computational tree $T_i(t)$ as follows:

1. Initially, set $y^{NEW} \leftarrow y^{MAP}$.
2. Update the value of the root variable of $T_i(t)$ by $y^{NEW}_i \leftarrow x^*_i$.
3. For each child factor ψ_α of root $i \in \alpha$, choose $\gamma \subset \alpha$ according to Condition *C3* and update the associated variable by $y^{NEW}_j \leftarrow x^*_j \ \ \forall j \in \gamma$.
4. Repeat Step 2,3 recursively by substituting $T_i(t)$ by the subtree of $T_i(t)$ of root $j \in \gamma$ until the process stops (i.e., $i = j$) or the leaf of $T_i(t)$ is reached (i.e., i does not have a child).

One can notice that the set of revised variables in Step 2 of the above procedure forms a path structure Q in the tree-structured GM. We first, consider the case that both ends of the path Q touch leaves of $T_i(t)$, where other cases can be argued in a similar manner. Define ζ_j and κ_j be the number of copies of x_j in path Q with $x^*_j = 1$ and $x^*_j = 0$, respectively, where $\zeta = [\zeta_j], \kappa = [\kappa_j] \in \mathbb{Z}^n_+$. Then, from our construction of y^{NEW}, one can observe that

$$\begin{aligned} y^{NEW} &= y^{MAP} + \zeta - \kappa \\ w \cdot y^{MAP} - w \cdot y^{NEW} &= w \cdot (\kappa - \zeta). \end{aligned}$$

If we set $z = x^* + \varepsilon(\kappa - \zeta)$ where $0 < \varepsilon < \min\{1/2t, \eta\}$, then one can check that z satisfies the conditions of Lemma 3 using Conditions *C2, C3*. Hence, from Lemma 3, there exists $z^\ddagger \in \mathcal{P}$ such that

$$\begin{aligned} &\|z^\ddagger - z\|_1 \le \varepsilon K \\ &\|z^\ddagger - x^*\|_1 \ge \varepsilon(\|\zeta\|_1 + \|\kappa\|_1 - K) \ge \varepsilon(t - K). \end{aligned}$$

where $z = x^* + \varepsilon(\kappa - \zeta)$. Hence, it follows that

$$\begin{aligned} 0 < \rho &\leq \frac{w \cdot z^\ddagger - w \cdot x^*}{\|z^\ddagger - x^*\|_1} \\ &\leq \frac{w \cdot z + \varepsilon w_{\max} K - w \cdot x^*}{\varepsilon(t-K)} \\ &= \frac{\varepsilon w \cdot (\kappa - \zeta) + \varepsilon w_{\max} K}{\varepsilon(t-K)} \\ &= \frac{w \cdot (\kappa - \zeta) + w_{\max} K}{t - K} \end{aligned}$$

Furthermore, if $t > \left(\frac{w_{\max}}{\rho} + 1\right) K$, the above inequality implies that

$$\begin{aligned} w \cdot y^{MAP} - w \cdot y^{NEW} &= w \cdot (\kappa - \zeta) \\ &\geq \rho t - (w_{\max} + \rho) K \ > \ 0. \end{aligned}$$

This contradicts to the fact that y^{MAP} is a MAP configuration. This completes the proof of Theorem 1.

3.2 PROOF OF LEMMA 3

One can write $\mathcal{P} = \{x \ : \ Ax \geq b\} \subset [0,1]^n$ for some matrix $A \in \mathbb{R}^{m \times n}$ and vector $b \in \mathbb{R}^m$, where without loss of generality, we can assume that $\|A_i\|_2 = 1$ where $\{A_i\}$ is the set of row vectors of A. We define

$$\mathcal{P}_\varepsilon = \{x \ : \ Ax \geq b - \varepsilon \mathbf{1}\},$$

where $\mathbf{1}$ is the vector of ones. Then, one can check that $z \in \mathcal{P}_\varepsilon$ for z, ε satisfying conditions of Lemma 3. Now we aim for finding a universal constant K satisfying

$$\texttt{dist}(\mathcal{P}, \mathcal{P}_\varepsilon) := \max_{x \in \mathcal{P}_\varepsilon} (\min_{y \in \mathcal{P}} \|x - y\|_1) \leq \varepsilon K,$$

which leads to the conclusion of Lemma 3.

To this end, for $\xi \subset [1, 2, \dots, m]$ with $|\xi| = n$, we let A_ξ be the square sub-matrix of A by choosing ξ-th rows of A and b_ξ is the n-dimensional subvector of b corresponding ξ. Throughout the proof, we only consider ξ such that A_ξ is invertible. Using this notation, we first claim the following.

Claim 4 *If A_ξ is invertible and $v_\xi := A_\xi^{-1} b_\xi \in \mathcal{P}$, then v_ξ is a vertex of polytope $\mathcal{P}$.*

Proof. Suppose v_ξ is not a vertex of $\mathcal{P}$, i.e. there exist $x, y \in \mathcal{P}$ such that $x \neq y$ and $v_\xi = \lambda x + (1-\lambda) y$ for some $\lambda \in (0, 1/2]$. Under the assumption, we will reach a contradiction. Since $\mathcal{P}$ is a convex set,

$$\frac{3\lambda}{2} x + \left(1 - \frac{3\lambda}{2}\right) y \in \mathcal{P}. \tag{8}$$

However, as A_ξ is invertible,

$$A_\xi \left(\frac{3\lambda}{2} x + \left(1 - \frac{3\lambda}{2}\right) y\right) \neq b_\xi. \tag{9}$$

From (8) and (9), there exists a row vector A_i of A_ξ and the corresponding element b_i of b_ξ such that

$$A_i \cdot \left(\frac{3\lambda}{2} x + \left(1 - \frac{3\lambda}{2}\right) y\right) > b_i.$$

Using the above inequality and $A_i \cdot (\lambda x + (1-\lambda) y) = b_i$, one can conclude that

$$A_i \cdot \left(\frac{\lambda}{2} x + \left(1 - \frac{\lambda}{2}\right) y\right) < b_i,$$

which contradict to $\frac{\lambda}{2} x + \left(1 - \frac{\lambda}{2}\right) y \in \mathcal{P}$. This completes the proof of Claim 4. □

We also note that if v is a vertex of polytope $\mathcal{P}$, there exists ξ such that A_ξ is invertible and $v = A_\xi^{-1} b_\xi$. We define the following notation:

$$\mathcal{I} = \{\xi \ : \ A_\xi^{-1} b_\xi \in \mathcal{P}\} \quad \mathcal{I}_\varepsilon = \{\xi \ : \ A_\xi^{-1}(b_\xi - \varepsilon \mathbf{1}) \in \mathcal{P}_\varepsilon\},$$

where Claim 4 implies that $\{v_\xi := A_\xi^{-1} b_\xi \ : \ \xi \in \mathcal{I}\}$ and $\{u_{\xi,\varepsilon} := A_\xi^{-1}(b_\xi - \varepsilon \mathbf{1}) \ : \ \xi \in \mathcal{I}_\varepsilon\}$ are sets of vertices of $\mathcal{P}$ and $\mathcal{P}_\varepsilon$, respectively. Using the notation, we show the following claim.

Claim 5 *There exists $\eta > 0$ such that $\mathcal{I}_\varepsilon \subset \mathcal{I}$ for all $\varepsilon \in (0, \eta)$.*

Proof. Suppose $\eta > 0$ satisfying the conclusion of Claim 5 does not exist. Then, there exists a strictly decreasing sequence $\{\varepsilon_k > 0 : k = 1, 2, \dots\}$ converges to 0 such that $\mathcal{I}_{\varepsilon_k} - \mathcal{I} \neq \emptyset$. Since $|\{\xi : \xi \subset [1, 2, \dots, m]\}| < \infty$, there exists ξ' such that

$$|\mathcal{K} := \{k \ : \ \xi' \in \mathcal{I}_{\varepsilon_k} - \mathcal{I}\}| = \infty. \tag{10}$$

For any $k \in \mathcal{K}$, observe that the sequence $\{u_{\xi', \varepsilon_\ell} \ : \ \ell \geq k, \ell \in \mathcal{K}\}$ converges to $v_{\xi'}$. Furthermore, all points in the sequence are in $\mathcal{P}_{\varepsilon_k}$ since $\mathcal{P}_{\varepsilon_\ell} \subset \mathcal{P}_{\varepsilon_k}$ for any $\ell \geq k$. Therefore, one can conclude that $v_{\xi'} \in \mathcal{P}_{\varepsilon_k}$ for all $k \in \mathcal{K}$, where we additionally use the fact that $\mathcal{P}_{\varepsilon_k}$ is a closed set. Because $\mathcal{P} = \bigcap_{k \in \mathcal{K}} \mathcal{P}_{\varepsilon_k}$, it must be that $v_{\xi'} \in \mathcal{P}$, i.e., $v_{\xi'}$ must be a vertex of $\mathcal{P}$ from Claim 4. This contradicts to the fact $\xi' \notin \mathcal{I}$. This completes the proof of Claim 5. □

From the above claim, we observe that any $x \in \mathcal{P}_\varepsilon$ can be expressed as a convex combination of $\{u_{\xi,\varepsilon} \ : \ \xi \in \mathcal{I}\}$, i.e., $x = \sum_{\xi \in I} \lambda_\xi u_{\xi,\varepsilon}$ with $\sum_{\xi \in I} \lambda_\xi = 1$ and $\lambda_\xi \geq 0$. For all $\varepsilon \in (0, \eta)$ for $\eta > 0$ in Claim 5, one can conclude that

$$\begin{aligned} \texttt{dist}(\mathcal{P}, \mathcal{P}_\varepsilon) &\leq \max_{x \in \mathcal{P}_\varepsilon} \|\sum_{\xi \in \mathcal{I}} \lambda_\xi u_{\xi,\varepsilon} - \sum_{\xi \in \mathcal{I}} \lambda_\xi v_\xi\|_1 \\ &= \max_{x \in \mathcal{P}_\varepsilon} \varepsilon \|\sum_{\xi \in \mathcal{I}} \lambda_\xi A_\xi^{-1} \mathbf{1}\|_1 \\ &\leq \varepsilon \max_\xi \|A_\xi^{-1} \mathbf{1}\|_1, \end{aligned}$$

where we choose $K = \max_\xi \|A_\xi^{-1} \mathbf{1}\|_1$. This completes the proof of Lemma 3.

4 APPLICATIONS OF THEOREM 1 TO COMBINATORIAL OPTIMIZATION

In this section, we introduce concrete instances of LPs satisfying the conditions of Theorem 1 so that BP correctly converges to its optimal solution. Specifically, we consider LP formulations associated to several combinatorial optimization problems including shortest path, maximum weight perfect matching, traveling salesman, maximum weight disjoint vertex cycle packing and vertex cover. We note that the shortest path result, Corollary 6, is known (Ruozzi and Tatikonda, 2008), where we rediscover it as a corollary of Theorem 1. Our other results, Corollaries 7-11, are new and what we first establish in this paper.

4.1 SHORTEST PATH

Given directed graph $G = (V, E)$ and non-negative edge weights $w = [w_e : e \in E] \in \mathbb{R}_+^{|E|}$, the shortest path problem is to find the shortest path from the source s to the destination t: it minimizes the sum of edge weights along the path. One can naturally design the following LP for this problem:

$$\begin{aligned} \text{minimize} \quad & w \cdot x \\ \text{subject to} \quad & \sum_{e \in \delta^o(v)} x_e - \sum_{e \in \delta^i(v)} x_e \\ & = \begin{cases} 1 & \text{if } v = s \\ -1 & \text{if } v = t \\ 0 & \text{otherwise} \end{cases} \quad \forall v \in V \\ & x = [x_e] \in [0,1]^{|E|}. \end{aligned} \tag{11}$$

where $\delta^i(v), \delta^o(v)$ are the set of incoming, outgoing edges of v. It is known that the above LP always has an integral solution, i.e., the shortest path from s to t. We consider the following GM for LP (11):

$$\Pr[X = x] \propto \prod_{e \in E} e^{-w_e x_e} \prod_{v \in V} \psi_v(x_{\delta(v)}), \tag{12}$$

where the factor function ψ_v is defined as

$$\psi_v(x_{\delta(v)}) = \begin{cases} 1 & \text{if } \sum_{e \in \delta^o(v)} x_e - \sum_{e \in \delta^i(v)} x_e \\ & \quad = \begin{cases} 1 & \text{if } v = s \\ -1 & \text{if } v = t \\ 0 & \text{otherwise} \end{cases} \\ 0 & \text{otherwise} \end{cases}.$$

For the above GM (12), one can easily check Conditions *C2, C3* of Theorem 1 hold and derive the following corollary whose formal proof is presented in the supplementary material due to the space constraint.

Corollary 6 *If the shortest path from s to t, i.e., the solution of the shortest path LP (11), is unique, then the max-product BP on GM (12) converges to it.*

The uniqueness condition in the above corollary is easy to guarantee by adding small random noises to edge weights.

4.2 MAXIMUM WEIGHT PERFECT MATCHING

Given undirected graph $G = (V, E)$ and non-negative edge weights $w = [w_e : e \in E] \in \mathbb{R}_+^{|E|}$ on edges, the maximum weight perfect matching problem is to find a set of edges such that each vertex is connected to exactly one edge in the set and the sum of edge weights in the set is maximized. One can naturally design the following LP for this problem:

$$\begin{aligned} \text{maximize} \quad & w \cdot x \\ \text{subject to} \quad & \sum_{e \in \delta(v)} x_e = 1, \quad \forall v \in V \\ & x = [x_e] \in [0,1]^{|E|}. \end{aligned} \tag{13}$$

where $\delta(v)$ is the set of edges connected to a vertex v. If the above LP has an integral solution, it corresponds to the solution of the maximum weight perfect matching problem.

It is known that the maximum weight matching LP (13) always has a half-integral solution $x^* \in \{0, \frac{1}{2}, 1\}^{|E|}$. We will design BP for obtaining the half-integral solution. To this end, duplicate each edge e to e_1, e_2 and define a new graph $G' = (V, E')$ where $E' = \{e_1, e_2 : e \in E\}$. Then, we suggest the following equivalent LP that always have an integral solution:

$$\begin{aligned} \text{maximize} \quad & w' \cdot x \\ \text{subject to} \quad & \sum_{e_i \in \delta(v)} x_{e_i} = 2 \quad \forall v \in V \\ & x = [x_{e_i}] \in [0,1]^{|E'|}. \end{aligned} \tag{14}$$

where $w'_{e_1} = w'_{e_2} = w_e$. One can easily observe that solving LP (14) is equivalent to solving LP (13) due to our construction of G' and w'. Now, construct the following GM for LP (14):

$$\Pr[X = x] \propto \prod_{e_i \in E'} e^{w'_{e_i} x_{e_i}} \prod_{v \in V} \psi_v(x_{\delta(v)}), \tag{15}$$

where the factor function ψ_v is defined as

$$\psi_v(x_{\delta(v)}) = \begin{cases} 1 & \text{if } \sum_{e_i \in \delta(v)} x_{e_i} = 2 \\ 0 & \text{otherwise} \end{cases}.$$

For the above GM (15), we derive the following corollary of Theorem 1 whose formal proof is presented in the supplementary material due to the space constraint.

Corollary 7 *If the solution of the maximum weight perfect matching LP (14) is unique, then the max-product BP on GM (15) converges it.*

Again, the uniqueness condition in the above corollary is easy to guarantee by adding small random noises to edge weights $[w'_{e_i}]$. We note that it is known (Bayati et al., 2011) that BP converges to the unique and integral solution of LP (13), while Corollary 7 implies that BP can solve it without the integrality condition. We note that one can easily obtain a similar result for the maximum weight (non-perfect) matching problem, where we omit the details in this paper.

4.3 MAXIMUM WEIGHT PERFECT MATCHING WITH ODD CYCLES

In previous section we prove that BP converges to the optimal (possibly, fractional) solution of LP (14), equivalently LP (13). One can add odd cycle (also called Blossom) constraints and make those LPs tight i.e. solves the maximum weight perfect matching problem:

$$\begin{aligned} \text{maximize} \quad & w \cdot x \\ \text{subject to} \quad & \sum_{e \in \delta(v)} x_e = 1, \quad \forall v \in V \\ & \sum_{e \in C} x_e \le \frac{|C|-1}{2}, \quad \forall C \in \mathcal{C}, \\ & x = [x_e] \in [0,1]^{|E|}. \end{aligned} \tag{16}$$

where $\mathcal{C}$ is a set of odd cycles in G. The authors (Shin et al., 2013) study BP for solving LP (16) by replacing $\sum_{e \in \delta(v)} x_e = 1$ by $\sum_{e \in \delta(v)} x_e \le 1$, i.e., for the maximum weight (non-perfect) matching problem. Using Theorem 1, one can extend the result to the maximum weight perfect matching problem, i.e., solving LP (16). To this end, we follow the approach (Shin et al., 2013) and construct the following graph $G' = (V', E')$ and weight $w' = [w'_e : e \in E'] \in \mathbb{R}^{|E'|}$ given set $\mathcal{C}$ of disjoint odd cycles:

$$V' = V \cup \{v_C : C \in \mathcal{C}\}$$
$$E' = \{(u, v_C) : u \in C, C \in \mathcal{C}\} \cup E \setminus \{e \in C : C \in \mathcal{C}\}$$

$$w'_e = \begin{cases} \frac{1}{2} \sum_{e' \in E(C)} (-1)^{d_C(u,e')} w_{e'} & \text{if } e = (u, v_C) \\ & \text{for some } C \in \mathcal{C}, \\ w_e & \text{otherwise} \end{cases}$$

where $d_C(u, e')$ is the graph distance between u, e' in cycle C. Then, LP (16) is equivalent to the following LP:

$$\begin{aligned} \text{maximize} \quad & w' \cdot y \\ \text{subject to} \quad & \sum_{e \in \delta(v)} y_e = 1, && \forall v \in V \\ & \sum_{u \in V(C)} (-1)^{d_C(u,e)} y_{(v_C,u)} \in [0,2], && \forall e \in E(C) \\ & \sum_{e \in \delta(v_C)} y_e \le |C| - 1, && \forall C \in \mathcal{C} \\ & y = [y_e] \in [0,1]^{|E'|}. \end{aligned} \tag{17}$$

Now, we construct the following GM from the above LP:

$$\Pr[Y = y] \propto \prod_{e \in E} e^{w_e y_e} \prod_{v \in V} \psi_v(y_{\delta(v)}) \prod_{C \in \mathcal{C}} \psi_C(y_{\delta(v_C)}), \tag{18}$$

where the factor function ψ_v, ψ_C is defined as

$$\psi_v(y_{\delta(v)}) = \begin{cases} 1 & \text{if } \sum_{e \in \delta(v)} y_e = 1 \\ 0 & \text{otherwise} \end{cases},$$

$$\psi_C(y_{\delta(v_C)}) = \begin{cases} 1 & \text{if } \sum_{u \in V(C)} (-1)^{d_C(u,e)} y_{(v_C,u)} \in \{0, 2\} \\ & \quad \sum_{e \in \delta(v_C)} y_e \le |C| - 1 \\ 0 & \text{otherwise} \end{cases}.$$

For the above GM (18), we derive the following corollary of Theorem 1 whose formal proof is presented in the supplementary material due to the space constraint.

Corollary 8 *If the solution of the maximum weight perfect matching with odd cycles LP (17) is unique and integral, then the max-product BP on GM (18) converges to it.*

We again emphasize that a similar result for the maximum weight (non-perfect) matching problem was established in (Shin et al., 2013). However, the proof technique in the paper does not extend to the perfect matching problem. This is in essence because presumably the perfect matching problem is harder than the non-perfect matching one. Under the proposed generic criteria of Theorem 1, we overcome the technical difficulty.

4.4 VERTEX COVER

Given undirected graph $G = (V, E)$ and non-negative integer vertex weights $b = [b_v : v \in V] \in \mathbb{Z}_+^{|V|}$, the vertex cover problem is to find a set of vertices minimizes the sum of vertex weights in the set such that each edge is connected to at least one vertex in it. This problem is one of Karp's 21 NP-complete problems (Karp, 1972). The associated LP formulation to the vertex cover problem is as follows:

$$\begin{aligned} \text{minimize} \quad & b \cdot y \\ \text{subject to} \quad & y_u + y_v \ge 1, (u, v) \in E \\ & y = [y_v] \in [0,1]^{|V|}. \end{aligned} \tag{19}$$

However, if we design a GM from the above LP, it does not satisfy conditions in Theorem 1. Instead, we will show that BP can solve the following dual LP:

$$\begin{aligned} \text{maximize} \quad & \sum_{e \in E} x_e \\ \text{subject to} \quad & \sum_{e \in \delta(v)} x_e \le b_v, \quad \forall v \in V \\ & x = [x_e] \in \mathbb{R}_+^{|E|}. \end{aligned} \tag{20}$$

Note that the above LP always has a half-integral solution. As we did in Section 4.2, one can duplicate edges, i.e.,

$E' = \{e_1, \dots, e_{2b_{\max}} : e \in E\}$ with $b_{\max} = \max_v b_v$, and design the following equivalent LP having an integral solution:

$$\begin{aligned} \text{maximize} \quad & w' \cdot x \\ \text{subject to} \quad & \sum_{e_i \in \delta(v)} x_{e_i} \le 2b_v, \quad \forall v \in V \\ & x = [x_{e_i}] \in [0,1]^{|E'|} \end{aligned}, \tag{21}$$

where $w'_{e_i} = w_e$ for $e \in E$ and its copy $e_i \in E'$. From the above LP, we can construct the following GM:

$$\Pr[X = x] \propto \prod_{e_i \in E'} e^{w'_{e_i} x_{e_i}} \prod_{v \in V} \psi_v(x_{\delta(v)}), \tag{22}$$

where the factor function ψ_v is defined as

$$\psi_v(x_{\delta(v)}) = \begin{cases} 1 & \text{if } \sum_{e_i \in \delta(v)} x_{e_i} \le 2b_v \\ 0 & \text{otherwise} \end{cases}.$$

For the above GM (22), we derive the following corollary of Theorem 1 whose formal proof is presented in the supplementary material due to the space constraint.

Corollary 9 *If the solution of the vertex cover dual LP (21) is unique, then the max-product BP on GM (22) converges it.*

Again, the uniqueness condition in the above corollary is easy to guarantee by adding small random noises to edge weights $[w'_{e_i}]$. We further remark that if the solution of the primal LP (19) is integral, then it can be easily found from the solution of the dual LP (21) using the strictly complementary slackness condition (Bertsimas and Tsitsiklis, 1997) .

4.5 TRAVELING SALESMAN

Given directed graph $G = (V, E)$ and non-negative edge weights $w = [w_e : e \in E] \in \mathbb{R}_+^{|E|}$, the traveling salesman problem (TSP) is to find the minimum weight Hamiltonian cycle in G. The natural LP formulation to TSP is following:

$$\begin{aligned} \text{minimize} \quad & w \cdot x \\ \text{subject to} \quad & \sum_{e \in \delta(v)} x_e = 2 \\ & x = [x_e] \in [0,1]^{|E|}. \end{aligned} \tag{23}$$

From the above LP, one can construct the following GM:

$$\Pr[X = x] \propto \prod_{e \in E} e^{-w_e x_e} \prod_{v \in V} \psi_v(x_{\delta(v)}), \tag{24}$$

where the factor function ψ_v is defined as

$$\psi_v(x_{\delta(v)}) = \begin{cases} 1 & \text{if } \sum_{e \in \delta(v)} x_e = 2 \\ 0 & \text{otherwise} \end{cases}.$$

For the above GM (24), we derive the following corollary of Theorem 1 whose formal proof is presented in the supplementary material due to the space constraint.

Corollary 10 *If the solution of the traveling salesman LP (23) is unique and integral, then the max-product BP on GM (24) converges it.*

Again, the uniqueness condition in the above corollary is easy to guarantee by adding small random noises to edge weights. In Section 5, we show the empirical performance of the max-product BP on GM (24) for solving TSP without relying on the integrality condition in Corollary 10.

4.6 MAXIMUM WEIGHT CYCLE PACKING

Given undirected graph $G = (V, E)$ and non-negative edge weights $w = [w_e : e \in E] \in \mathbb{R}_+^{|E|}$, the maximum weight vertex disjoint cycle packing problem is to find the maximum weight set of cycles with no common vertex. It is easy to observe that it is equivalent to find a subgraph maximizing the sum of edge weights on it such that each vertex of the subgraph has degree 2 or 0. The natural LP formulation to this problem is following:

$$\begin{aligned} \text{maximize} \quad & w \cdot x \\ \text{subject to} \quad & \sum_{e \in \delta(v)} x_e = 2y_v \\ & x = [x_e] \in [0,1]^{|E|}, y = [y_v] \in [0,1]^{|V|}. \end{aligned} \tag{25}$$

From the above LP, one can construct the following GM:

$$\Pr[X = x, Y = y] \propto \prod_{e \in E} e^{w_e x_e} \prod_{v \in V} \psi_v(x_{\delta(v)}, y_v), \tag{26}$$

where the factor function ψ_v is defined as

$$\psi_v(x_{\delta(v)}, y_v) = \begin{cases} 1 & \text{if } \sum_{e \in \delta(v)} x_e = 2y_v \\ 0 & \text{otherwise} \end{cases}.$$

For the above GM (26), we derive the following corollary of Theorem 1 whose formal proof is presented in the supplementary material due to the space constraint.

Corollary 11 *If the solution of maximum weight vertex disjoint cycle packing LP (25) is unique and integral, then the max-product BP on GM (26) converges it.*

Again, the uniqueness condition in the above corollary is easy to guarantee by adding small random noises to edge weights.

5 EXPERIMENTAL RESULTS FOR TRAVELING SALESMAN PROBLEM

In this section, we report empirical performances of BP on GM (24) for solving the traveling salesman problem (TSP)

Table 1: Experimental results for small size complete graph and each number is the average among 100 samples. For example, Greedy+BP means that the Greedy algorithm using edge weights as BP beliefs as we describe in Section 5. The left value is the approximation ratio, i.e., the average weight ratio between the heuristic solution and the exact solution. The right value is the average weight of the heuristic solutions. The last row is a ratio of tight TSP LP (23).

Size	5	10	15	20	25
Greedy	1.07 / 1.84	1.20 / 2.25	1.33 / 2.58	1.51 / 2.85	1.51 / 3.04
Greedy+BP	1.00 / 1.75	1.05 / 1.98	1.13 / 2.23	1.19 / 2.27	1.21 / 2.43
Christofides	1.38 / 1.85	1.38 / 2.56	1.67 / 3.20	1.99 / 3.75	2.16 / 4.32
Christofides+BP	1.00 / 1.75	1.09 / 2.07	1.23 / 2.43	1.30 / 2.50	1.45 / 2.90
Insertion	1.03 / 1.79	1.29 / 2.38	1.53 / 2.95	1.72 / 3.26	1.89 / 3.77
Insertion+BP	1.00 / 1.75	1.29 / 2.39	1.52 / 2.97	1.79 / 3.38	1.94 / 3.89
N-Neighbor	1.07 / 1.84	1.27 / 2.39	1.42 / 2.74	1.55 / 2.96	1.64 / 3.30
N-Neighbor+BP	1.00 / 1.75	1.05 / 1.98	1.13 / 2.23	1.15 / 2.21	1.20 / 2.40
2-Opt	1.00 / 1.75	1.08 / 2.04	1.12 / 2.21	1.24 / 2.36	1.28 / 2.57
2-Opt+BP	1.00 / 1.75	1.04 / 1.96	1.07 / 2.11	1.11 / 2.13	1.16 / 2.34
Tight LPs	100%	93%	88%	87%	84%

Table 2: Experimental results for sparse Erdos-Renyi graph with fixed average vertex degrees and each number is the average among 1000 samples. The left value is the ratio that a heuristic finds the Hamiltonian cycle without penalty edges. The right value is the average weight of the heuristic solutions.

Size	100			200		
Degree	10	25	50	10	25	50
Greedy	0% / 7729.43	0.3% / 2841.98	13% / 1259.08	0% / 15619.9	0% / 5828.88	0.3% / 2766.07
Greedy+BP	14% / 1612.82	21% / 1110.27	44% / 622.488	6.4% / 2314.95	10% / 1687.29	16% / 1198.48
Christoifeds	0% / 19527.3	0% / 16114.3	0% / 10763.7	0% / 41382.5	0% / 37297.0	0% / 32023.1
Christofides+BP	14% / 2415.73	20% / 1663.47	34% / 965.775	6.1% / 3586.77	9.2% / 2876.35	12% / 2183.80
Insertion	0% / 12739.2	84% / 198.099	100% / 14.2655	0% / 34801.6	0.9% / 3780.71	99% / 44.1293
Insertion+BP	0% / 13029.0	76% / 283.766	100% / 14.6964	0% / 34146.7	0.3% / 4349.11	99% / 41.2176
N-Neighbor	0% / 9312.77	0% / 3385.14	7.6% / 1531.83	0% / 19090.7	0% / 7383.23	0.3% / 3484.82
N-Neighbor+BP	16% / 1206.95	26% / 824.232	50% / 509.349	6.9% / 1782.17	12% / 1170.38	24% / 888.421
2-Opt	34% / 1078.03	100% / 14.6873	100% / 7.36289	2% / 3522.78	100% / 35.8421	100% / 18.6147
2-Opt+BP	76% / 293.450	100% / 13.5773	100% / 6.53995	33% / 1088.79	100% / 34.7768	100% / 17.4883
Tight LPs	62%	62.3%	63%	52.2%	55%	52.2%

that is presumably the most famous one in combinatorial optimization. In particular, we design the following BP-guided heuristic for solving TSP:

1. Run BP for a fixed number of iterations, say 100, and calculate the BP marginal beliefs (3).

2. Run the known TSP heuristic using edge weights as $\log \frac{b[0]}{b[1]}$ using BP margianl beliefs instead of the original weights.

For TSP heuristic in Step 2, we use Greedy, Christoifeds, Insertion, N-Neighbor and 2-Opt provided by the LEMON graph library (Dezső et al., 2011). We first perform the experiments on the complete graphs of size 5, 10, 15, 20, 25 and random edge weight in $(0, 1)$ to measure approximation qualities of heuristics, where it is reported in Table 1. Second, we consider the sparse Erdos-Renyi random graph of size 100, 200 and random edge weight in $(0, 1)$. Then, we make it a complete graph by adding non-existing edges with penalty edge weight 1000.[2] For these random instances, we report performance of various heuristics in Table 2. Our experiments show that BP boosts performances of known TSP heuristics in overall, where BP is very easy to code and does not hurt the simplicity of heuristics.

[2]This is to ensure that a Hamiltonian cycle always exists.

6 CONCLUSION

The BP algorithm has been the most popular algorithm for solving inference problems arising graphical models, where its distributed implementation, associated ease of programming and strong parallelization potential are the main reasons for its growing popularity. In this paper, we aim for designing BP algorithms solving LPs, and provide sufficient conditions for its correctness and convergence. We believe that our results provide new interesting directions on designing efficient distributed (and parallel) solvers for large-scale LPs.

Acknowledgements.

We would like to acknowledge the support of the AOARD project, FA2386-14-1-4058.

References

Mohsen Bayati, Devavrat Shah, and Mayank Sharma. Maximum weight matching via max-product belief propagation. In *Information Theory, 2005. ISIT 2005. Proceedings. International Symposium on*, pages 1763–1767. IEEE, 2005.

Mohsen Bayati, Christian Borgs, Jennifer Chayes, and Riccardo Zecchina. Belief propagation for weighted b-matchings on arbitrary graphs and its relation to linear programs with integer solutions. *SIAM Journal on Discrete Mathematics*, 25(2):989–1011, 2011.

Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA, 1997.

Venkat Chandrasekaran, Nathan Srebro, and Prahladh Harsha. Complexity of inference in graphical models. In *UAI 2008, Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence*, pages 70–78. AUAI Press, 2008.

George B Dantzig. *Linear programming and extensions*. Princeton university press, 1998.

Balázs Dezső, Alpár Jüttner, and Péter Kovács. Lemon–an open source c++ graph template library. *Electronic Notes in Theoretical Computer Science*, 264(5):23–45, 2011.

Joseph Gonzalez, Yucheng Low, and Carlos Guestrin. Parallel splash belief propagation. Technical report, DTIC Document, 2010.

Bert C Huang and Tony Jebara. Loopy belief propagation for bipartite maximum weight b-matching. In *International Conference on Artificial Intelligence and Statistics*, pages 195–202, 2007.

Richard M Karp. *Reducibility among combinatorial problems*. Springer, 1972.

Leonid G Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980.

Yucheng Low, Joseph Gonzalez, Aapo Kyrola, Danny Bickson, Carlos Guestrin, and Joseph M. Hellerstein. Graphlab: A new framework for parallel machine learning. In *UAI 2010, Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, pages 340–349. AUAI Press, 2010.

Nam Ma, Yinglong Xia, and Viktor K Prasanna. Task parallel implementation of belief propagation in factor graphs. In *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*, pages 1944–1953. IEEE, 2012.

Marc Mezard and Andrea Montanari. *Information, physics, and computation*. Oxford University Press, 2009.

Tom Richardson and Ruediger Urbanke. *Modern coding theory*. Cambridge University Press, 2008.

Nicholas Ruozzi and Sekhar Tatikonda. st paths using the min-sum algorithm. In *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*, pages 918–921. IEEE, 2008.

Justin Salez and Devavrat Shah. Belief propagation: an asymptotically optimal algorithm for the random assignment problem. *Mathematics of Operations Research*, 34 (2):468–480, 2009.

Sujay Sanghavi, Dmitry Malioutov, and Alan Willsky. Belief propagation and lp relaxation for weighted matching in general graphs. *Information Theory, IEEE Transactions on*, 57(4):2203–2212, 2011.

Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer, 2003.

Jinwoo Shin, Andrew E Gelfand, and Misha Chertkov. A graphical transformation for belief propagation: Maximum weight matchings and odd-sized cycles. In *Advances in Neural Information Processing Systems*, pages 2022–2030, 2013.

George B Dantzig Mukund N Thapa. Linear programming. 2003.

Meritxell Vinyals, Alessandro Farinelli, Juan A Rodríguez-aguilar, et al. Worst-case bounds on the quality of max-product fixed-points. In *Advances in Neural Information Processing Systems*, pages 2325–2333, 2010.

Martin J Wainwright and Michael I Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2): 1–305, 2008.

Yair Weiss. Belief propagation and revision in networks with loops. 1997.

Yair Weiss and William T Freeman. On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. *Information Theory, IEEE Transactions on*, 47(2):736–744, 2001.

Jonathan S Yedidia, William T Freeman, and Yair Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *Information Theory, IEEE Transactions on*, 51(7):2282–2312, 2005.

Practical Message-passing Framework for Large-scale Combinatorial Optimization

Inho Cho* Soya Park* Sejun Park Dongsu Han Jinwoo Shin

KAIST

***Abstract*—Graphical Model (GM) has provided a popular framework for big data analytics because it often lends itself to distributed and parallel processing by utilizing graph-based 'local' structures. It models correlated random variables where in particular, the max-product Belief Propagation (BP) is the most popular heuristic to compute the most-likely assignment in GMs. In the past years, it has been proven that BP can solve a few classes of combinatorial optimization problems under certain conditions.**

Motivated by this, we explore the prospect of using BP to solve generic combinatorial optimization problems. The challenge is that, in practice, BP may converge very slowly and even if it does converge, the BP decision often violates the constraints of the original problem. This paper proposes a generic framework that enables us to apply BP-based algorithms to compute an approximate feasible solution for an arbitrary combinatorial optimization task. The main novel ingredients include (a) careful initialization of BP messages, (b) hybrid damping on BP updates, and (c) post-processing using BP beliefs. Utilizing the framework, we develop parallel algorithms for several large-scale combinatorial optimization problems including maximum weight matching, vertex cover and independent set. We demonstrate that our framework delivers high approximation ratio, speeds up the process by parallelization, and allows large-scale processing involving billions of variables.

***Keywords*-Combinatorial optimization, Belief propagation, Parallel algorithm, Maximum weighted matching**

I. Introduction

Graphical Models (GMs) provide a useful framework for modeling and processing real-world, large-scale data applications. From traditional big data analytics, such as page rank [1] and graph mining [2], to more recent deep learning [3], graphical models have been commonly applied for processing large-scale data-sets. GM is a particularly good fit for big data applications because it lends itself to fast parallel implementations by utilizing graph-based 'local' structures. Several modern programming models, such as GraphLab [4], GraphChi [5] and GraphX [6], enable distributed, parallel computation on GMs.

One of the most common computational tasks found in GM's applications is to compute the most-likely assignment to random variables, so-called a MAP (Maximum-A-Posteriori) estimate. This can be viewed as solving a large-scale optimization problem, which is becoming increasingly important for big data analytics since it presents a major computational bottleneck. Motivated by this, we explore the prospect of using GMs to solve optimization problems at scale. In particular, we propose a message-passing based algorithm to solve combinatorial optimization problems based on Belief Propagation (BP). The (max-product) BP algorithm is a well-studied heuristic that has been popularly used to approximately solve MAP optimization tasks. It is an iterative, message-passing algorithm proven to produce exact solutions for tree structured GMs. However, understanding on the performance of BP for loopy GMs has been quite limited.

Our goal is to design a highly accurate approximation algorithm based on BP that solves generic large-scale combinatorial optimization problems. The benefit of such an algorithm is that it inherently lends itself to parallel implementations, enabling fast performance and ensuring scalability. However, the challenge is that the BP algorithm is not guaranteed to be correct or even converge in general. Even if it converges to the correct solution, its convergence speed is too low for solving large-scale instances. Especially, when there are multiple optima (multiple solutions), BP is known to oscillate in many cases [7, 8, 9]. One can stop BP iterations without waiting for convergence, but then BP algorithms often produce infeasible solutions, i.e., violate the constraints of the targeted combinatorial optimization.

Contribution. We resolve the issues by designing a generic BP-based framework that computes highly accurate and feasible approximation solutions. The basic idea is to use a truncated BP algorithm in conjunction with existing heuristics to enforce a feasible solution in a way that ensures high approximation ratio. At a high level, the algorithm takes the following steps. First, given an optimization problem, we represent the problem in the MAP framework of GM. Second, we run the corresponding BP's message-passing and 'partially' solve the optimization problem. However, because BP often does not converge quickly, we run only a fixed number of BP iterations; we do not wait for convergence. Instead, to boost up the quality of BP decisions, we (a) carefully initialize the BP messages, (b) add a small noise to weights, and (c) apply a hybrid damping strategy on BP message updates (Section III-B). Finally, we apply existing heuristics as post-processing procedures to enforce the feasibility of the BP decision. In particular, we run known heuristics for a given combinatorial optimization problem by replacing the parameters of the original problems (e.g., edge weights) with BP beliefs. This ensures that the framework is applicable to any combinatorial optimization problems,

*The first two authors contributed equally to this work.

978-1-4799-9926-2/15/$31.00 ©2015 IEEE

while achieving a higher approximation ratio than existing heuristics.

In summary, this paper makes three key contributions:

1) **Practical BP-based algorithm design:** To the best of our knowledge, this paper is the first to propose a generic concept for designing BP-based algorithms that solve large-scale combinatorial optimization problems.
2) **Parallel implementation:** We also demonstrate that the algorithm is easily parallelizable. For the maximum weighted matching problem, this translates to 71x speed up while sacrificing only 0.1% accuracy compared to the state-of-art exact algorithm [10].
3) **Extensive empirical evaluation:** We evaluate our algorithms on three different combinatorial optimization problems on diverse synthetic and real-world data-sets. Our evaluation shows that the framework shows higher accuracy compared to other known heuristics.

Related Work. In the past years, the convergence and correctness of BP has been studied analytically for several classical combinatorial optimization problems, including matchings [7, 11, 12], perfect matchings [13], shortest paths [8], independent sets [14], network flows [9] and vertex covers [15]. The important common feature of these models is that BP converges to a correct assignment when the linear programming (LP) relaxation of the combinatorial optimization is tight, i.e., when it shows no integrality gap. However, LP tightness is an inevitable condition to guarantee the convergence of BP to the optimal solution, which is the main limitation of these theoretical studies towards wider applicability. Moreover, even if BP converges to the optimal solution, its convergence speed is often too slow for solving large-scale instances. There have been also empirical studies of BP-based algorithms for specific combinatorial optimization instances, including traveling salesman [16], graph partitioning [16], Steiner tree [17] and network alignment [18]. However, their focuses are not on large-scale instances and the running times of the proposed algorithms typically grow super-linearly with respect to the input size. In contrast, we provide a generic framework on designing BP-based scalable, parallel algorithms that are widely applicable to arbitrary large-scale combinatorial optimization problems.

Organization. We provide backgrounds on BP and combinatorial optimization problems in Section II. Section III describes our BP-based algorithm design, and Section IV provides details on its parallel implementation. We evaluate our algorithm on several combinatorial optimization problems in Section V. Finally, we conclude in Section VI.

II. Preliminaries

A. Graphical Model and Belief Propagation

A joint distribution of n (binary) random variables $Z = [Z_i] \in \{0,1\}^n$ is called a Graphical Model (GM) if it factorizes as follows: for $z = [z_i] \in \{0,1\}^n$,

$$\Pr[Z = z] \propto \prod_{i\in\{1,\ldots,n\}} \psi_i(z_i) \prod_{\alpha\in F} \psi_\alpha(z_\alpha),$$

where $\{\psi_i, \psi_\alpha\}$ are non-negative functions, so-called factors; F is a collection of subsets

$$F = \{\alpha_1, \alpha_2, ..., \alpha_k\} \subset 2^{\{1,2,\ldots,n\}}$$

(each α_j is a subset of $\{1, 2, \ldots, n\}$ with $|\alpha_j| \geq 2$); z_α is the projection of z onto dimensions included in α.[1] In particular, ψ_i is called a variable factor. Assignment z^* is called a maximum-a-posteriori (MAP) assignment if $z^* = \arg\max_{z\in\{0,1\}^n} \Pr[z]$. This means that computing a MAP assignment requires us to compare $\Pr[z]$ for all possible z, which is typically computationally intractable (i.e., NP-hard) unless the induced bipartite graph of factors F and variables z, the so-called factor graph, has a bounded tree width [19].

The max-product belief propagation (BP) is a popular heuristic for approximating the MAP assignment in GM. BP is implemented iteratively; at each iteration t, BP maintains four messages $\{m^t_{\alpha\to i}(c), m^t_{i\to\alpha}(c) : c \in \{0,1\}\}$ between every variable z_i and every associated $\alpha \in F_i$, where $F_i := \{\alpha \in F : i \in \alpha\}$. The messages are updated as follows:

$$m^{t+1}_{\alpha\to i}(c) = \max_{z_\alpha : z_i = c} \psi_\alpha(z_\alpha) \prod_{j\in\alpha\setminus i} m^t_{j\to\alpha}(z_j) \quad (1)$$

$$m^{t+1}_{i\to\alpha}(c) = \psi_i(c) \prod_{\alpha'\in F_i\setminus\alpha} m^t_{\alpha'\to i}(c). \quad (2)$$

One can reduce the complexity of messages by combining (1) and (2) as:

$$m^{t+1}_{i\to\alpha}(c) = \psi_i(c) \prod_{\alpha'\in F_i\setminus\alpha} \max_{z_{\alpha'} : z_i = c} \psi_{\alpha'}(z_{\alpha'}) \prod_{j\in\alpha'\setminus i} m^t_{j\to\alpha'}(z_j).$$

Given a set of messages $\{m_{i\to\alpha}(c), m_{\alpha\to i}(c) : c \in \{0,1\}\}$, the so-called BP marginal beliefs are computed as follows:

$$b_i[z_i] = \psi_i(z_i) \textstyle\prod_{\alpha\in F_i} m_{\alpha\to i}(z_i). \quad (3)$$

This BP algorithm outputs $z^{BP} = [z^{BP}_i]$ where

$$z^{BP}_i = \begin{cases} 1 & \text{if } b_i[1] > b_i[0] \\ ? & \text{if } b_i[1] = b_i[0] \\ 0 & \text{if } b_i[1] < b_i[0] \end{cases}.$$

It is known that z^{BP} converges to a MAP assignment after a sufficient number of iterations, if the factor graph is a tree and the MAP assignment is unique. However, if the graph contains cycles or MAP is not unique, the BP algorithm is not guaranteed to converge to a MAP assignment in general.

B. Belief Propagation for Combinatorial Optimization

The max-product BP can be applied to compute an approximate solution for any 'discrete' optimizations. This

[1] For example, if $z = [0, 1, 0]$ and $\alpha = \{1, 3\}$, then $z_\alpha = [0, 0]$.

section describes the maximum weight matching problem as an example. Given a graph $G = (V, E)$ and edge weights $w = [w_e] \in \mathbb{R}^{|E|}$, it finds a set of edges such that each vertex is connected to at most one edge in the set and the sum of edge weights in the set is maximized. The problem is formulated as the following IP (Integer Programming):

$$\begin{aligned} &\text{maximize} \quad w \cdot x \\ &\text{s.t.} \quad \sum_{e \in \delta(v)} x_e \leq 1, \ \forall v \in V, \quad x = [x_e] \in \{0,1\}^{|E|}, \end{aligned} \tag{4}$$

where $\delta(v)$ is the set of edges incident to vertex $v \in V$.

Now consider the following GM: for $x = [x_e] \in \{0,1\}^{|E|}$,

$$\Pr[X = x] \ \propto \ \prod_{e \in E} e^{w_e x_e} \prod_{v \in V} \psi_v(x_{\delta(v)}), \tag{5}$$

where $$\psi_v(x_{\delta(v)}) = \begin{cases} 1 & \text{if } \sum_{e \in \delta(v)} x_e \leq 1 \\ 0 & \text{otherwise} \end{cases}.$$

One can easily observe that the MAP assignments for GM (5) correspond to the (optimal) solution of IP (4). Therefore, one can use the max-product BP for solving the maximum weight matching problem, where the algorithm can be simplified using $a_{i \to j} = \log \frac{m_{i \to (i,j)}(0)}{m_{i \to (i,j)}(1)}$ as described in Algorithm 1.

Algorithm 1 BP for Maximum Weight Matching

(ITERATION) Calculate new messages as follows:

$$a_{i \to j}^{t+1} \leftarrow \max_{k \in \delta(i) \setminus \{j\}} \left\{ \max \left\{ w_{ik} - a_{k \to i}^{t}, 0 \right\} \right\}$$

(DECISION) For each edge $e = (i, j) \in E$, decide

$$x_e \ = \ \begin{cases} 1 & \text{if} \quad (a_{i \to j} + a_{j \to i}) < w_e \\ ? & \text{if} \quad (a_{i \to j} + a_{j \to i}) = w_e \\ 0 & \text{if} \quad (a_{i \to j} + a_{j \to i}) > w_e \end{cases}.$$

Note that one can design similar BP algorithms for arbitrary combinatorial optimization problems (e.g., minimum weight vertex cover and maximum weight independent set) as we demonstrate in Section V.

III. Algorithm Design

The main goal of this paper is to design BP-based parallel algorithms for solving combinatorial optimizations. To this end, one can design a BP algorithm as described in Section II-B. However (a) it might diverge or converge very slowly, (b) even if it converges quickly, the BP decision might be not correct, and (c) even worse, BP might produce an infeasible solution, i.e., it does not satisfy the constraints of the problem.

To address these issues, we propose a generic BP-based framework that provides highly accurate approximate solutions for combinatorial optimization problems. The framework has two steps, as shown in Figure 1. In the first phase, it runs a BP algorithm for a fixed number of iterations without waiting for convergence. Then, the second phase runs a known heuristic using BP beliefs instead of the original weights to output a feasible solution. Namely, the first and second phases are respectively designed for 'BP weight transforming' and 'post-processing'. In the following sections, we describe the two phases in more detail in reverse order. For illustration purposes, we focus on the maximum weight matching problem while the results are derived using Erdős–Rényi random graphs with 1000 vertices, average degree 100, and edge weights drawn from the uniform distribution over the interval $[0, 1]$. Later in Section V, we demonstrate the framework is applicable to any other combinatorial optimization problems.

A. Post-Processing Phase.

The decision on BP beliefs might give an infeasible solution. For example, in the maximum weight matching problem, BP decides $x_e = 0, 1$ based on the sign of $w_{ij} - (a_{i \to j} + a_{j \to i})$ (see Algorithm 1), but the edges of $x_e = 1$ might not form a matching even if BP converges, i.e., there exists a vertex v such that $\sum_{e \in \delta(v)} x_e > 1$.

To resolve the issue, we use post-processing by utilizing existing heuristics to the given problem that find a feasible solution. Applying post-processing ensures that the solution is at least feasible. In addition, our key idea is to replace the original weights by the logarithm of BP beliefs, i.e., the new weight on edge e becomes: $w_{ij} - (a_{i \to j} + a_{j \to i})$. After this, we apply known heuristics using the logarithm of BP beliefs to achieve higher accuracy.

For example, the following 'local' greedy algorithm can be used as a post-processing mechanism:

1. Initially, all vertices are 'unmatched', i.e., $x_e = 0$ for all $e \in E$.
2. Choose an arbitrary unmatched vertex i and match it to an unmatched vertex $j \neq i$ having the highest value in $w_{ij} - (a_{i \to j} + a_{j \to i})$, i.e., set $x_{ij} = 1$.
3. Keep iterating step 2 until no more vertex can be matched.

To confirm the effectiveness of the proposed post-processing mechanism, we compare it with the following two alternative post-processing schemes that remove edges to enforce matching after BP processing in a naive manner:

- Random: If there exists a vertex v such that $\sum_{e \in \delta(v)} x_e > 1$ on the BP decision, randomly select one edge and remove other edges.
- Weight: If there exists a vertex v such that $\sum_{e \in \delta(v)} x_e > 1$ on the BP decision, remove edges of smaller weight.

Figure 3(a) compares the approximation ratio obtained using BP-belief-based post-processing versus the naive post-processing heuristics (random and weight). It shows that the proposed BP-belief-based post-processing outperforms the rest. Note, the results in Figure 3 were obtained by first applying BP message passing for weight transformation. Next, we explain how this is done in our framework.

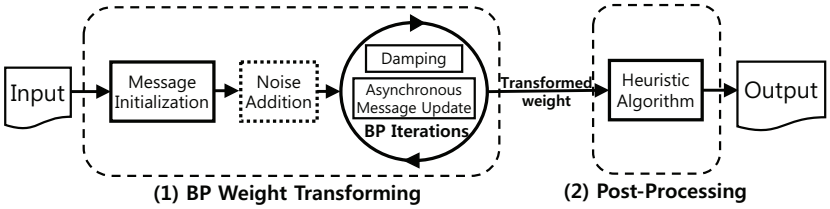

Figure 1: Overview of our generic BP-based framework

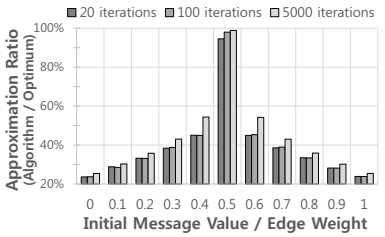

Figure 2: Effects of initial messages on the number of BP iterations. We set $a^0_{i\to j} = a^0_{j\to i} = c \cdot w_{ij}$ for a value c of x-axis.

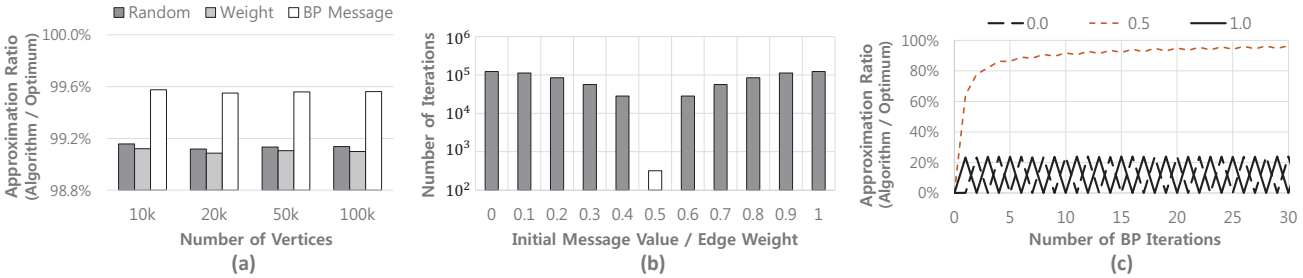

Figure 3: (a) Average approximation ratio for different post-processing schemes. We use a local greedy algorithm as a post-processing based on original weights and BP messages (i.e., beliefs). The 'Random selection' post-processing is also compared. (b) Effects of initial messages on the convergence of BP. We set $a^0_{i\to j} = a^0_{j\to i} = c \cdot w_{ij}$ for the value c of x-axis (c) Approximation ratio for different initial messages $a^0_{i\to j} = a^0_{j\to i} = 0, w_{ij}/2, w_{ij}$

B. BP Weight Transforming Phase

To improve the approximation quality and solve the convergence issues, we use three modifications to the standard BP algorithm: (1) careful initialization on messages, (2) noise addition and (3) hybrid damping on message updates.

Message Initialization. The standard message initialization is $m^0_{\alpha\to i} = m^0_{i\to\alpha} = 1$, i.e., $a^0_{i\to j} = 0$ for the maximum weight matching problem (see Algorithm 1). The convergence rate of BP depends on the initialized messages. As reported in Figure 3(b), we try different initializations, $a^0_{i\to j} = c \cdot w_{ij}$ for $0 \le c \le 1$, where the case $c = 0.5$ shows the fastest convergence. The main intuition we found for explaining such a phenomenon is as follows. If $a^0_{i\to j} = w_{ij}/2$, the BP decision at the initial step is neutral, i.e., x_e =?, since $a^0_{i\to j} + a^0_{j\to i} = w_{ij}$. On the other hand, if $a^0_{i\to j} = 0$, BP chooses $x_e = 1$ initially for all edges and most likely does $x_e = 0$ for most edges in the next step, i.e., it keeps oscillating between $x_e = 1$ and $x_e = 0$ for a while. The choice $a^0_{i\to j} = w_{ij}/2$ alleviates the fluctuation behavior of BP and boosts up its convergence speed.

We remind that, under our framework, BP runs only for a fixed number of iterations since it might converge too slowly, even with the initialization $a^0_{i\to j} = w_{ij}/2$, for practical purposes. With fixed number of iterations, careful initialization becomes even more critical as experimental results in Figure 3(c) and Figure 2 suggest. For example, if one runs 5000 iterations of BP, they show that the standard initialization ($a^0_{i\to j} = 0$) achieves at most 30% approximation ratio, while the proposed method ($a^0_{i\to j} = w_{ij}/2$) achieves 99%. Moreover, one can also observe that the advantage of more BP updates diminishes as the number of iterations becomes large. The observation holds for much larger graphs. Thus, we only run 100 BP iterations in our algorithm and do not wait for BP's convergence.

# vertices (# edges)	Approximation Ratio		Difference
	Multiple optima	Unique optimum	
1k (50k)	99.88 %	99.90 %	-0.02 %
5k (250k)	99.86 %	99.85 %	+0.01 %
10k (500k)	99.85 %	99.84 %	+0.01 %
20k (1M)	99.84 %	99.83 %	+0.01 %

Table I: Approximation ratio of BP for MWM with multiple optima and a unique optimum. We introduce a small noise to the edge weights and set the initial message by $a^0_{i\to j} = a^0_{j\to i} = w_{ij}/2$.

Noise Addition. The BP algorithm often oscillates when the MAP solution is not unique. To address this issue, we transform the original problem to one that has a unique solution with high probability by adding small noises to the weights, i.e., $w_e \leftarrow w_e + r_e$, where $r_e \in [-r, r]$ is a random number chosen independently across edges. We apply this to all cases. Here, one has to be careful in deciding the range r of noises. If r_e is too large, the quality of BP solution deteriorates because the optimal solution might have changed from the original problem. On the other hand, if r_e is too small compared to w_e, BP converges very slowly. To achieve a balance, we choose the range r of noise r_e as 10% of the minimum distance among weights. We find that this results in over 99.8% approximation ratio even when the solution is not unique, which has little difference with that of unique solution as shown in Table I.

Hybrid Damping. To boost up the convergence speed of BP updates further, we use a specific damping strategy to alleviate message oscillation. We update messages to be the

# vertices (# edges)	Approximation Ratio			
	no-damp(100)	damp(100)	no-damp(50) +damp(50)	damp(50) +no-damp(50)
10k (500k)	99.58 %	99.69 %	**99.83 %**	99.56 %
20k (1M)	99.55 %	99.68 %	**99.82 %**	99.56 %
50k (2.5M)	99.56 %	99.69 %	**99.83 %**	99.57 %
100k (5M)	99.56 %	99.69 %	**99.83 %**	99.57 %

Table II: Approximation ratio of BP without damping, BP with damping, BP with damping only for first 50 iterations, and BP with damping for last 50 iterations. We introduce a small noise to the edge weights and set the initial message by $a^0_{i\to j} = a^0_{j\to i} = w_{ij}/2$.

average of old and new messages.

$$a^{t+1}_{i\to j} \leftarrow \max_{k\in\delta(i)\setminus\{j\}} \left\{\max\left\{w_{ik} - a^t_{k\to i}, 0\right\}\right\}$$
$$a^{t+1}_{i\to j} \leftarrow (a^t_{i\to j} + a^{t+1}_{i\to j})/2.$$

We note that the damping strategy provides a similar effect as our proposed initialization $a^0_{i\to j} = w_{ij}/2$. Hence, if one uses both, the effect of one might be degraded due to the other. Due to this, we first run the half of BP iterations without damping (this is for keeping the effect of the proposed initialization) and perform the last half of BP iterations with damping. As reported in Table II, this hybrid approach outperforms other alternatives, including (a) no use of damping, (b) using damping in every iteration, and (c) damping in the first half of BP iterations and no-damping in the last half.

IV. Parallel Design and Implementation

This section addresses issues in parallelization of our algorithm. First, we introduce asynchronous message update that enables efficient parallelization of BP message passing. Second, we illustrate the issues in parallelizing post-processing. Finally, we describe the parallel implementations of our algorithm and their benefits.

A. Asynchronous Message Update

So far, we have assumed that there is only one thread, and BP messages are updated synchronously among vertices after calculating new message values. Thus, each iteration consists of two phases: message calculation phase and message update phase.

For parallelization, we first divide the graph by partitioning the vertices, and assign each partition to a single thread (see Section IV-C for details). However, if we naively parallelize the process using multiple threads, frequent synchronization may incur large overhead. Thus, we apply asynchronous message update where each vertex updates the message value right after new message value is calculated and eliminate synchronization point between iterations. This makes the process faster because of the reduced synchronization points. Figure 4 shows that performance improvement (speed up in running time) of asynchronous update over synchronous is up to 237% in our example graph for the maximum weight matching problem with 16 threads[2]; we leave detailed evaluation in Section V.

We now discuss its impact on approximation quality. Two factors marginally affect the approximation quality in opposite directions. First, updating the message value of each vertex right after its message calculation marginally improves the approximation ratio, as shown in Figure 5. Updating a message right after its calculation on a individual vertex basis implicitly has a similar effect to applying an additional iteration, which improves the quality. Second, having multiple threads run without synchronizing across iterations marginally degrade the approximation quality. The reason is that some threads run faster than others, and messages from the slower threads are not updated as frequently. We quantify this effect in Figure 5 and find that the impact is marginal; we still achieve 99.9% accuracy with multiple threads.

In summary, using asynchronous message updates, we speed up the run-time of the algorithm by up to 240%, while achieving 99.9% approximation ratio. In Section V, we show that the results also extend to other combinatorial optimization problems.

B. Local Post-Processing

The second phase of our algorithm runs existing heuristics for post-processing to enforce the feasibility of BP decisions. While the framework works with *any* heuristics-based post-processing methods, for the entire process to be parallel, it is important that the post-processing step is also parallel. An important criterion for efficient parallelization is locality of computation; if the post-processing heuristics can compute the result locally without requiring global knowledge, they can be easily parallelized. Moreover, if they do not require synchronization, the running time can be further reduced.

Fortunately, for most combinatorial optimization problems heuristics that match the two criteria exist. A local greedy algorithm, for example, enables local post-processing (i.e., it does not require global information). and require little synchronization. As reported in Section V, we also evaluate our algorithm in conjunction with other post-processing heuristics to demonstrate its flexibility.

C. Parallel Implementation

The BP algorithm is easy to parallelize because of its message passing nature. To demonstrate this, we parallelize our BP-based framework using three platforms:

- *GraphChi* enables large-scale graph computation on a personal computer by utilizing disk drive [5]. Using its API, we specify the BP message update rules to enable parallel message updates and scale the size of the problem up to billions of variables.
- *OpenMP* is a task-based parallelization library developed by Intel. Simply putting OpenMP pragma directive enables the compiler to support parallel computation.
- For our *pthread*-based implementation, we divide a single BP iteration into the smaller execution blocks

[2]We use a machine with two Intel Xeon(R) CPU E5-2690 @ 2.90GHz each with 8 cores.

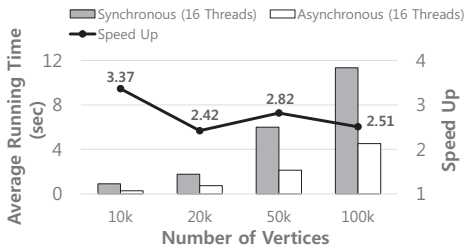

Figure 4: Average running time of our BP-based algorithm with synchronous message update and asynchronous message update. We apply all three modifications to BP of Section III-B.

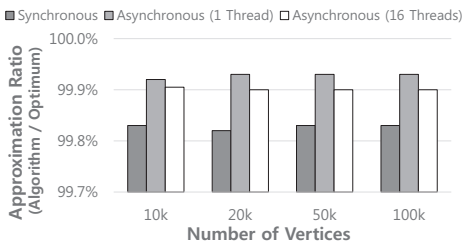

Figure 5: Approximation ratio of our BP-based algorithm with synchronous and asynchronous message update with 16 threads. We apply all three modifications to BP of Section III-B.

	# Vertices	# Edges
apache1	80k	230k
apache2	715k	2M
ecology2	1M	2M
G3_circuit	1.6M	3M
bone010	1M	23.4M

Table III: Summary of MWM data-sets

	# Vertices	Optimal Cost	
		MVC	MIS
frb-30-15	450	420	30
frb-45-21	945	900	45
frb-53-24	1,272	1,219	53
frb-59-26	1,534	1,475	59

Table IV: Summary of MWVC and MWIS data-sets

called tasks. Each iteration is divided into per-thread tasks. Because we have fixed multiple number of BP iterations, it concerns many more tasks. We put these tasks in a task queue. Initially all threads are assigned a task and the thread finished its task will be assigned the next task in the task queue. This minimizes the overlap between different iterations and synchronization points, which reduces the run time while obtaining high approximation ratio.

Algorithm 2 BP for Minimum Weight Vertex Cover

(ITERATION) Calculate new messages as follows:

$$a_{i\to j}^{t+1} \leftarrow \min\left\{ w_v + \sum_{k\in\delta(i)\setminus\{j\}} a_{k\to i}^{t}, 0 \right\}$$

(DECISION) For each vertex $i \in V$, decide

$$x_i \;=\; \begin{cases} 1 & \text{if} \quad \sum_{j\in\delta(i)} a_{j\to i} < -w_i \\ ? & \text{if} \quad \sum_{j\in\delta(i)} a_{j\to i} = -w_i \\ 0 & \text{if} \quad \sum_{j\in\delta(i)} a_{j\to i} > -w_i \end{cases}$$

Algorithm 3 BP for Maximum Weight Independent Set

(ITERATION) Calculate new messages as follows:

$$a_{i\to j}^{t+1} \leftarrow \max\left\{ w_v - \sum_{k\in\delta(i)\setminus\{j\}} a_{k\to i}^{t}, 0 \right\}$$

(DECISION) For each vertex $i \in V$, decide

$$x_i \;=\; \begin{cases} 1 & \text{if} \quad \sum_{j\in\delta(i)} a_{j\to i} < w_i \\ ? & \text{if} \quad \sum_{j\in\delta(i)} a_{j\to i} = w_i \\ 0 & \text{if} \quad \sum_{j\in\delta(i)} a_{j\to i} > w_i \end{cases}$$

V. Evaluation

We evaluate our BP framework using three popular combinatorial optimization problems: maximum weight matching, minimum weight vertex cover and maximum weight independent set problem. We perform extensive empirical evaluation to demonstrate the benefit of our algorithm for the following three questions:

1) *Does the BP-based algorithm provide high approximation ratio?*
2) *Can the algorithm achieve speed-up due to parallel implementations?*
3) *Can it solve large-scale problems involving billions of variables?*

We already introduced the IP formulation of the maximum weight matching (MWM) in (4), where those of the minimum weight vertex cover (MWVC) and maximum weight independent set problem (MWIS) are as follows:

$$\begin{aligned} \textbf{MWVC:}\quad & \text{minimize} \quad w \cdot x \\ & \text{subject to} \quad x_u + x_v \geq 1,\ \forall e = (u,v) \in E \\ & \qquad x = [x_v] \in \{0,1\}^{|V|} \end{aligned} \tag{6}$$

$$\begin{aligned} \textbf{MWIS:}\quad & \text{maximize} \quad w \cdot x \\ & \text{subject to} \quad x_u + x_v \leq 1,\ \forall e = (u,v) \in E \\ & \qquad x = [x_e] \in \{0,1\}^{|V|}. \end{aligned} \tag{7}$$

We provide descriptions of BP algorithms in Algorithm 2–3 for MWVC and MWIS. As we propose in Section III-B, we choose initial BP messages for neutral decisions: $a_{j\to i}^{0} = -w_{ij}/|\delta(i)|$ for vertex cover and $a_{j\to i}^{0} = w_{ij}/|\delta(i)|$ for independent set.

A. Experiment Setup

In our experiments, both real-world and synthetic data-sets are used for evaluation. For MWM, we used data-sets from the university of Florida sparse matrix collection [20] summarized in Table III. For larger scale synthetic evaluation, we generate Erdős-Rényi random graphs (up to 50 million vertices with 2.5 billion edges) with average vertex degree of 100 with edge weights drawn independently from the uniform random distribution over the interval $[0, 1]$. For MWVC and MWIS, we use the frb-series from BHOSLIB [21] summarized in Table IV, where it also contains the optimal solutions. We note that we perform no experiment using synthetic data-sets for MWVC and MWIS since they are NP-hard problems, i.e., impossible to compute the optimal solutions. On the other hand, for MWM the Edmonds' Blossom algorithm [10] can compute the optimal solution in polynomial time. All experiments in this section are conducted on a machine with Intel Xeon(R) CPU E5-2690 @ 2.90GHz with 8 cores and 8 hyperthreads with 128GB of memory, unless otherwise noted.

		Approximation Ratio	
		Serial BP (1 thread)	Parallel BP (16 threads)
Synthetic # vertices (# edges)	500k (25M)	99.93 %	99.90 %
	1M (50M)	99.93 %	99.90 %
	2M (100M)	99.94 %	99.91 %
	5M (250M)	99.93 %	99.90 %
Real-world Data-set (Florida)	apache1	100.0 %	100.0 %
	apache2	100.0 %	100.0 %
	ecology2	100.0 %	100.0 %
	G3_circuit	99.95 %	99.95 %
	bone010	99.11 %	99.12 %

Table V: MWM: Approximation ratio of our BP-based algorithm on synthetic and sparse matrix collection data-sets [20].

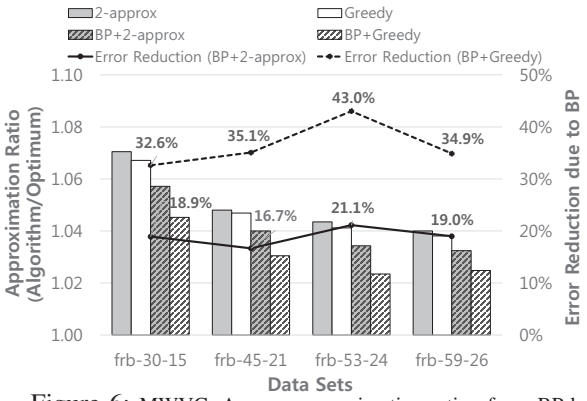

Figure 6: MWVC: Average approximation ratio of our BP-based algorithm, the 2-approximation algorithm and the greedy algorithm on frb-series data-sets.

B. Approximation Ratio

We now demonstrate our BP-based approximation algorithm produces highly accurate results. In particular, we show that our BP-based algorithms outperform well-known heuristics for MWVC, MWIS and closely approximate exact solutions for MWM for all cases we evaluate.

Maximum Weight Matching. For MWM, we compare the approximation qualities of serial, synchronous BP (i.e., one thread) in Section III and parallel, asynchronous implementation (i.e., using multiple threads) in Section IV on both synthetic and real-world data-sets, where we compute the optimal solution using the Blossom algorithm [10] to measure the approximation ratios. Table V summarize our experimental results for MWM for the synthetic data-sets and the Florida data. Our BP-based algorithm achieves 99% to 99.9% approximation ratios.

Minimum Weight Vertex Cover. For MWVC, we use two post-processing procedures: greedy and 2-approximation algorithm [22]. For the local greedy algorithm, we choose a random edge and add one of its adjacent vertices with a smaller weight until all edges are covered. We compare the approximation qualities of our BP-based algorithm compared to the cases when one uses only the greedy algorithm and the 2-approximation algorithm. Figure 6 shows the experimental results for the two post-processing heuristics. The results show that our BP-based weight transformation enhances the approximation quality of known approximation heuristics by up to 43%.

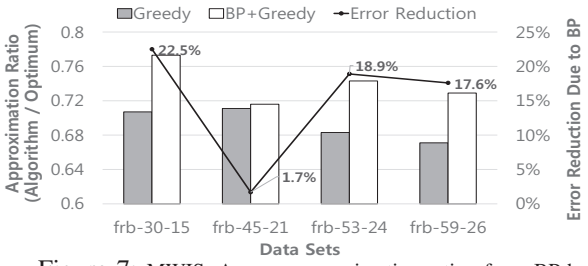

Figure 7: MWIS: Average approximation ratio of our BP-based algorithm and the greedy algorithm on frb-series data-sets.

Maximum Weight Independent Set. For MWIS, the experiment was performed on frb-series data-sets. We use a greedy algorithm as the post-processing procedure, which selects vertices in the order of higher weights until no vertex can be selected without violating the independent set constraint. We compare the approximation qualities of our BP-based algorithm and the standard greedy algorithm. Figure 7 shows that our BP-based framework enhances the approximation ratio of the solution by 2% to 23%.

C. Parallelization Speed-up

One of the important advantages of our BP-based algorithm is that it is fast, while delivering high approximation guarantees. In this section, we focus on the speed-up due to parallelization. Figure 8 compares the running time of the Blossom algorithm and our BP-based algorithm with 1 single core and 16 cores. With five million vertices, our asynchronous parallel implementation is eight times faster than the synchronous serial implementation, while still retaining 99.9% approximation ratio as reported in Table V. To demonstrate the overall benefit in context, we compare its running time with that of the current fastest implementation of the Blossom algorithm due to Kolmogorov [10]. Here, we note that the Blossom algorithm is inherently not easy to parallelize. For parallel implementation, we report results for our pthread implementation , but the OpenMP implementation also show comparable performance. For 20 million vertices (one billion edges), it shows that the running time of our algorithm can be accelerated by up to 71 times than the Blossom algorithm, while sacrificing 0.1% of accuracy. The running time gap is expected be more significant for larger graphs since the running times of our algorithm and the Blossom algorithm are linear and cubic with respect to the number of vertices, respectively. We also experiments our algorithms for other problems to demonstrate the parallelization speedup. Due to the space limitation, we only report that for the minimum weight vertex cover problem in Figure 9.

D. Large-scale Optimization

Our algorithm can also handle large-scale instances because it is based on GMs that inherently lend itself to parallel and distributed implementations. To demonstrate this, we create a large-scale instance containing up to 50 million vertices and 2.5 billion edges. We experiment our algorithm

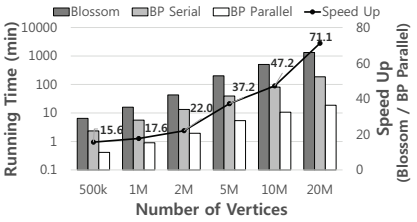

Figure 8: MWM: Running time of Blossom algorithm and our BP-based algorithms.

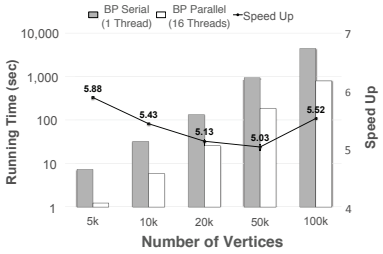

Figure 9: MWVC: Running time of our parallel BP-based algorithm (pthread implementation) on large-scale graphs.

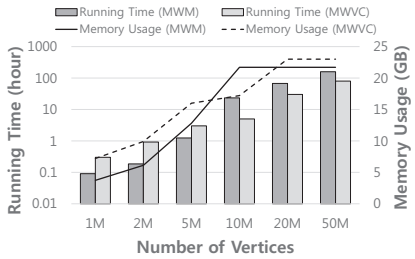

Figure 10: MWM and MWVC: Running time and memory usage of GraphChi-based implementation on large-scale graphs.

using GraphChi on a single consumer level machine with i7 CPU and 24GB of memory. Figure 10 shows the running time and memory usage of our algorithm for MWM and MWVC on large data-sets. For large graphs, GraphChi partitions them to load parts of them in memory, while storing the rest on disk by leveraging graph-based 'local' structures. Thus, we were able to solve problems with 2.5 billion edges on a single machine. In contrast, Kolmogorov's implementation [10] of the Blossom algorithm cannot handle such large graphs because distributed processing is difficult (e.g., it cannot handle more than 6M vertices on the same machine). Similarly, our algorithm can be run on multiple machines to scale to even larger problems. However, we leave this as future work.

VI. Conclusion

This paper explores the possibility of applying the BP algorithm to solve generic combinatorial optimizations at scale. We propose BP-based algorithm that achieves high approximation ratio and allows parallel implementation. We evaluate the algorithm's effectiveness and performance by applying our framework on three popular combinatorial optimization problems. Our evaluation shows that the algorithm outperforms existing approximation algorithms across many instances and is able to solve large-scale problems with billions of variables. We believe our BP-based framework is of broader interest for a wider class of large-scale optimization tasks.

Acknowledgment

This work was supported in part by Institute for Information & Communications Technology Promotion (IITP) granted by the Korea government (MSIP) (No.B0126-15-1078, Creation of PEP based on automatic protocol behavior analysis and Resource management for hyper connected for IoT Services); the Center for Integrated Smart Sensors funded by MSIP as Global Frontier Project (CISS-2011-0031863); and Asian Office of Aerospace Research and Development (AOARD) Project (FA2386-14-1-4058).

References

[1] S. Brin and L. Page, "Reprint of: The anatomy of a large-scale hypertextual web search engine," *Computer networks*, vol. 56, no. 18, pp. 3825–3833, 2012.

[2] D. Chakrabarti and C. Faloutsos, "Graph mining: Laws, generators, and algorithms," *ACM Computing Surveys (CSUR)*, vol. 38, no. 1, p. 2, 2006.

[3] R. Salakhutdinov and G. E. Hinton, "Deep boltzmann machines," in *International Conference on Artificial Intelligence and Statistics*, 2009, pp. 448–455.

[4] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein, "Distributed graphlab: A framework for machine learning and data mining in the cloud," *Proc. VLDB Endow.*, vol. 5, no. 8, pp. 716–727, Apr. 2012. [Online]. Available: http://dx.doi.org/10.14778/2212351.2212354

[5] A. Kyrola, G. E. Blelloch, and C. Guestrin, "Graphchi: Large-scale graph computation on just a pc." in *OSDI*, vol. 12, 2012, pp. 31–46.

[6] J. E. Gonzalez, R. S. Xin, A. Dave, D. Crankshaw, M. J. Franklin, and I. Stoica, "Graphx: Graph processing in a distributed dataflow framework," in *Proceedings of OSDI*, 2014, pp. 599–613.

[7] S. Sanghavi, D. Malioutov, and A. Willsky, "Belief propagation and lp relaxation for weighted matching in general graphs," *Information Theory, IEEE Transactions on*, vol. 57, no. 4, pp. 2203–2212, 2011.

[8] N. Ruozzi and S. Tatikonda, "st paths using the min-sum algorithm," in *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*. IEEE, 2008, pp. 918–921.

[9] D. Gamarnik, D. Shah, and Y. Wei, "Belief propagation for min-cost network flow: Convergence and correctness," *Operations Research*, vol. 60, no. 2, pp. 410–428, 2012.

[10] V. Kolmogorov, "Blossom v: a new implementation of a minimum cost perfect matching algorithm," *Mathematical Programming Computation*, vol. 1, no. 1, pp. 43–67, 2009.

[11] M. Bayati, D. Shah, and M. Sharma, "Maximum weight matching via max-product belief propagation," in *Information Theory, 2005. ISIT 2005. Proceedings. International Symposium on*. IEEE, 2005, pp. 1763–1767.

[12] B. C. Huang and T. Jebara, "Loopy belief propagation for bipartite maximum weight b-matching," in *International Conference on Artificial Intelligence and Statistics*, 2007, pp. 195–202.

[13] M. Bayati, C. Borgs, J. Chayes, and R. Zecchina, "Belief propagation for weighted b-matchings on arbitrary graphs and its relation to linear programs with integer solutions," *SIAM Journal on Discrete Mathematics*, vol. 25, no. 2, pp. 989–1011, 2011.

[14] S. Sanghavi, D. Shah, and A. S. Willsky, "Message passing for maximum weight independent set," *Information Theory, IEEE Transactions on*, vol. 55, no. 11, pp. 4822–4834, 2009.

[15] S. Park and J. Shin, "Max-product belief propagation for linear programming: Convergence and correctness," *arXiv preprint arXiv:1412.4972*, 2014.

[16] S. Ravanbakhsh, R. Rabbany, and R. Greiner, "Augmentative message passing for traveling salesman problem and graph partitioning," in *Advances in Neural Information Processing Systems*, 2014, pp. 289–297.

[17] M. Bayati, C. Borgs, A. Braunstein, J. Chayes, A. Ramezanpour, and R. Zecchina, "Statistical mechanics of steiner trees," *Physical review letters*, vol. 101, no. 3, p. 037208, 2008.

[18] M. Bayati, M. Gerritsen, D. F. Gleich, A. Saberi, and Y. Wang, "Algorithms for large, sparse network alignment problems," in *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*. IEEE, 2009, pp. 705–710.

[19] V. Chandrasekaran, N. Srebro, and P. Harsha, "Complexity of inference in graphical models," in *UAI 2008, Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence*. AUAI Press, 2008, pp. 70–78.

[20] T. A. Davis and Y. Hu, "The university of florida sparse matrix collection," *ACM Transactions on Mathematical Software (TOMS)*, vol. 38, no. 1, p. 1, 2011.

[21] "bhoslib benchmark set." [Online]. Available: http://iridia.ulb.ac.be/~fmascia/maximum_clique/BHOSLIB-benchmark

[22] C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.